مجله علمی-پژوهشی رایانش نرم و فن‌آوری اطلاعات

جلد ۵، شماره ۳، پاییز سال ۱۳۹۵ شاپا: ۱۰۰۶-۲۳۸۳

رهیافتی ترکیبی مبتنی بر الگوریتم بهینه‌سازی کلونی مورچه‌ها و اتوماتای یادگیر سلولی در حل مساله زمان‌بندی ایستای کارها در سیستم‌های چندپردازنده‌ای همگن

حمیدرضا بویری*

* نویسنده مسئول: آموزشکده فنی و حرفه ای سما، دانشگاه آزاد اسلامی، واحد شوشتر، شوشتر، ایران
boveiri@{shoushtar-samacollege.ir or ymail.com}

چکیده- زمان‌بندی کارها یکی از بزرگترین چالش‌ها در سیستم‌های چندپردازنده‌ای مانند سیستم‌های موازی و توزیع‌شده است. در این‌گونه سیستم‌ها هر برنامه حین کامپایل به قطعات کوچکتری به نام کار شکسته می‌شود. کارها مستقل نیستند و قیود اولویت (تقدم و تاخر) بین آنها جریان دارد. بدین ترتیب، زمان لازم جهت اجرای کارها، قیود اولویت بین کارها و هزینه‌های ارتباطی بین آنها با استفاده از یک گراف جهت‌دار غیرحلقوی به نام گراف وظایف مدلسازی می‌شود. کارهای یک برنامه باید به تعداد از پیش مشخصی پردازنده به گونه‌ای نگاشت شوند که قیود اولویت بین کارها رعایت شده و زمان اتمام کل کارها (خاتمه برنامه) حداقل شود. این مساله از جمله مسایل بغرنج زمانی (NP-hard) بوده و به‌دست آوردن بهترین زمان‌بندی ممکن با افزایش ابعاد مساله عموماً غیرممکن است؛ لذا اعمال روش‌های اکتشافی و فوق‌اکتشافی مختلف جهت حل این مساله و در راستایِ یافتن جواب‌های شبه‌بهینه منطقی است. دو فاکتور اصلی، طول زمان‌بندی به‌دست‌آمده از رهیافت‌های مختلف ارائه‌شده جهت حل این مساله را تحت شعاع قرار می‌دهد. اول اینکه کارها به چه ترتیبی جهت اجرا انتخاب شوند (زیرمساله ترتیب) و دوم اینکه ترتیب انتخاب‌شده چگونه بر روی پردازنده‌ها پخش شود (زیرمساله انتساب). در رهیافت پیشنهادی، الگوریتم بهینه‌سازی کلونی مورچه‌ها ترتیب اجرای کارها را مشخص کرده و اتوماتای یادگیر سلولی، ترتیب مشخص‌شده را روی پردازنده‌ها نگاشت می‌کند. جهت ارزیابی قسمت اول الگوریتم از شش گراف وظایف از برنامه‌های واقعی استفاده می‌شود که الگوریتم بهینه‌سازی کلونی مورچه‌ها در تمامی موارد قادر به یافتن ترتیب اجرای بهینه‌تری نسبت به روش‌های سنتی موجود است. در قسمت دوم الگوریتم نیز نتایج به‌دست‌آمده از اتوماتای یادگیر سلولی بهبود محسوسی نسبت به تنها رقیب سنتی خود یعنی روش کمترین زمان شروع ممکن (EST) دارد. در نهایت جهت ارزیابی عادلانه از ۱۲۵ گراف وظایف تصادفی با پارامترهای ساختاری مختلف استفاده شده که نتایج حاکی از آن است که رهیافت پیشنهادی از نظر عملکرد در هر دو زمینه بسیار موفق‌تر از الگوریتم‌های سنتیِ موجود بوده و در نهایت از این روش‌ها پیشی می‌گیرد.

واژه‌های کلیدی: زمان‌بندی ایستای کارها، سیستم‌های موازی و توزیع‌شده، گراف وظایف، الگوریتم بهینه‌سازی کلونی مورچه‌ها، اتوماتای یادگیر سلولی.

۱- مقدمه

امروزه با توجه به افزایش پیچیدگی زمانی برنامه‌ها و کاهش قیمت سخت‌افزار، استفاده از سیستم‌های چندپردازنده‌ای[1] چون سیستم‌های موازی و توزیع‌شده[2] رو به افزایش است. در این‌گونه سیستم‌ها معمولاً هر برنامه به قطعات کوچکتری به نام کار[3] شکسته می‌شود. کارها به سادگی می‌توانند پروسیجرهای[4] تشکیل‌دهنده‌ی یک برنامه باشند. به عبارت دیگر در حالت کلی می‌توان گفت هر برنامه از تعدادی پروسیجر تشکیل شده پس هر برنامه را می‌توان به تعدادی کار تقسیم کرد. کارها لزوماً مستقل نیستند. برخی کارها جهت اجرا به داده‌های تولیدشده توسط سایر کارها محتاج‌اند و بدین ترتیب مساله تقدم و تاخر بین کارها بوجود می‌آید. جهت فرموله‌بندی و نمایش مساله، زمان لازم جهت اجرای هر کار[5]، قیود اولویت بین کارها[6] و هزینه‌های انتقال داده بین

[1] Multiprocessor
[2] Parallel and Distributed Systems
[3] Task
[4] Procedure
[5] Execution Time
[6] Precedence Constrains

کارها[1] را با استفاده از یک گراف جهت‌دار غیر حلقوی[2] به نام گراف گراف وظایف[3] مدلسازی می‌کنند. همچنین کارها باید جهت اجرا به تعدادی پردازنده به گونه‌ای منتسب شوند که علاوه بر آنکه اولویت بین کارها رعایت شود زمان اتمام برنامه (زمان اتمام کل کارها)[4] نیز حداقل گردد. در حالتی که تمامی پردازنده‌ها از نظر قدرت پردازشی یکسان باشند زمانبندی را همگن[5] و در غیر اینصورت زمانبندی را غیر همگن[6] گویند. در صورتی که زمان اجرای کارها، قیود اولویت و هزینه‌های ارتباطی در حین مرحله کامپایل برنامه مشخص شوند آنگاه زمانبندی ایستا[7] یا معین[8] و در غیراینصورت زمانبندی پویا[9] یا نامعین[10] است. این مساله از جمله مسایل بغرنج زمانی[11] بوده و یافتن بهترین زمانبندی ممکن در حالت کلی و با افزایش ابعاد مساله، غیرممکن است [۶]. لذا روش‌های اکتشافی[12] و فوق‌اکتشافی[13] مختلفی جهت حل این مساله ارائه شده است که سهم زیادی از این تجربات متعلق به رهیافت‌های تکاملی چون الگوریتم ژنتیک است [۱]-[۵]. بطور کلی روش‌های معرفی‌شده در این زمینه را می‌توان به سه دسته اصلی تقسیم کرد [۷] و [۸]:

- دسته اول روش‌هایی که مبتنی بر تکثیر کارها[14] بوده مانند PY [۹]، DSH [۱۰]، LWB [۱۱]، BTDH [۱۲]، LCTD [۱۳]، CPFD [۱۴]، MJD [۱۵] و DFRN [۱۶] که تکثیر کارها روی پردازنده‌ها را مجاز می‌دانند تا با حذف کردن هزینه‌های ارتباطی بین یک کار و فرزندانش، زمان اتمام برنامه را کوتاهتر کنند. دو موضوع مهم طراحی اینگونه الگوریتم‌ها را بسیار پیچیده کرده است. اول اینکه کدام کارها تکثیر شوند و دوم اینکه روی کدامین پردازنده‌ها تکثیر انجام شود تا زمانبندی بهتری بدست آید. این الگوریتم‌ها همچنین حافظه بیشتری مصرف می‌کنند و البته در سیستم‌های واقعی امکان تکثیر و اجرای مجدد برخی کارها مانند تراکنش‌های بانکی وجود ندارد. لذا سایر روش‌های موجود که کاربرد فراوانتری دارند تکثیر کارها روی پردازنده‌ها را مجاز نمی‌دانند.
- دسته بعدی به نام خوشه‌های نامحدود[15] شامل روش‌هایی چون LC [۱۷]، EZ [۱۸]، MD [۱۹]، DSC [۲۰] و DSP [۲۱] می‌باشد که زمانبندی را روی تعداد نامحدودی پردازنده () می‌سازند و اغلب از روش خوشه‌بندی گراف وظایف استفاده می‌کنند. بدین معنا که ابتدا هر کار را یک خوشه در نظر می‌گیرند سپس با استفاده از معیارهایی دو خوشه را انتخاب می‌کنند اگر ادغام این دو خوشه زمانبندی را طولانی نکند آنها را ادغام کرده و در غیر اینصورت به سراغ دو خوشه دیگر می‌روند. این عمل تا جایی تکرار می‌شود که هیچ دو خوشه‌ای را نتوان ادغام کرد. در نهایت هر خوشه را جهت اجرا به یک پردازنده منتسب می‌کنند و زمانبندی کلی بدست می‌آید. می‌دانیم که فرض بر نامحدود بودن یا دلخواه بودن تعداد پردازنده‌ها نامعقول است. لذا این دسته از روش‌ها نیز کاربرد کمتری در مسائل واقعی دارند.
- دسته سوم که کاربرد بیشتری داشته و رهیافت پیشنهادی نیز از همین گروه می‌باشد؛ روش‌هایی به نام پردازنده‌های محدود[16] مانند HLFET [۲۲]، ISH [۱۰]، CLANS [۲۳]، LAST [۲۴]، ETF [۲۵]، DLS [۲۶] وMCP [۱۹] می‌باشند که تعداد پردازنده‌ها از پیش مشخص و محدود است و اغلب از روش زمانبندی لیستی[17] استفاده می‌کنند. در این روش با استفاده از معیارهایی[18]، لیستی از کارها به ترتیب اولویت ساخته می‌شود. سپس مرحله به مرحله، پراولویت‌ترین کار درون لیست خارج شده و به پردازنده‌ای که زودتر می‌تواند آن را اجرا کند اختصاص می‌یابد. تا جائیکه کل کارها از لیست خارج شوند و زمانبندی نهایی ایجاد شود. دو عامل اصلی، طول زمانبندی بدست‌آمده از رهیافت‌های این دسته را تحت شعاع قرار می‌دهد. اول اینکه کارها به چه ترتیبی جهت اجرا انتخاب شوند (مساله ترتیب) و دوم این مساله که ترتیب انتخاب‌شده جهت اجرا چگونه به پردازنده‌ها نگاشت شود (مساله انتساب).

در این مقاله از الگوریتم فوق‌اکتشافی کلونی مورچه‌ها[19] جهت یافتن ترتیب مناسب اجرای کارها استفاده می‌شود. رهیافت فوق‌اکتشافی کلونی مورچه‌ها یک الگوریتم موازی است که در آن یک کلونی از مورچه‌های مصنوعی جهت یافتن بهینه‌ترین مسیرهای حل مساله با یکدیگر همکاری می‌کنند. در حقیقت یک سیستم چندعامله است که در آن عامل‌ها (مورچه‌های مصنوعی)

[1] Communication Cost
[2] Directed Acyclic Graph
[3] Task Graph
[4] Finish-Time or Makespan
[5] Homogeneous
[6] Heterogeneous
[7] Static Scheduling
[8] Deterministic
[9] Dynamic Scheduling
[10] Undeterministic
[11] NP-hard
[12] Heuristic
[13] Meta-heuristic
[14] Task Duplication Based (TDB)
[15] Unbounded Number of Clusters (UNC)
[16] Bounded Number of Processors (BNP)
[17] List Scheduling
[18] Priority Measurements
[19] Ant Colony Optimization

در تعامل با هم با استفاده از یک ارتباط غیر مستقیم محلی به نام استیگمرجی[1] سعی در حل مساله دارند. این رهیافت با موفقیت بر روی مسائل گسسته بغرنجی چون فروشنده دوره‌گرد[2]، مساله انتساب درجه دوم[3]، مساله زمانبندی کارهای کارخانه[4]، مساله مسیریابی خودروها[5]، مرتب‌سازی ترتیبی گراف[6]، مسیریابی در شبکه‌های ارتباطی[7] و ... اعمال شده و در برخی موارد بهترین نتایج را نسبت به سایر الگوریتم‌های فوق‌اکتشافی داده است [۲۸].

الگوریتم‌های مبتنی بر زمانبندی لیستی (همچون رهیافت پیشنهادی) هر چند با استفاده از اولویت‌های مختلف، ترتیب‌های اجرای مناسبی می‌یابند اما در نهایت زمانبندی بدست‌آمده متاثر از روش انتساب ترتیب کارها به پردازنده‌هاست. اغلب از روش سنتی انتساب کار به پردازنده‌ای که کمترین زمان اجرای ممکن را فراهم می‌کند استفاده می‌شود[8]. که به نظر می‌رسد بهینه نبوده و روش‌های زمانبندی در این دسته را محدود می‌کند. لذا در این مقاله ابتدا بهینه‌نبودن این روش نشان داده شده و در ادامه رهیافتی پیشنهادی مبتنی بر اتوماتای یادگیر سلولی ارائه می‌شود که می‌تواند انتساب بهتری انجام دهد.

اتوماتای سلولی [۲۹] مدلی ریاضی جهت بررسی رفتار سیستم‌هایی است که در آنها مولفه‌های ساده‌ای با همکاری یکدیگر الگوهای پیچیده‌ای تولید می‌کنند. یک اتوماتای سلولی از یک شبکه منظم از سلول‌ها تشکیل شده که هر سلول در هر لحظه می‌تواند در یکی از k حالت متفاوت خود باشد. سلول‌های اتوماتای سلولی در زمان‌های گسسته بطور همزمان طبق یک قانون محلی بهنگام می‌شوند که در آن، مقدار جدید هر سلول بر اساس مقادیر سلول‌های همسایه تعیین می‌شود. اتوماتای سلولی بر اساس معیارهای مورد بررسی مانند بُعد شبکه، مقادیر مختلف k، مرز همسایگی و ... به دسته‌های مختلفی تقسیم می‌شوند.

اتوماتای یادگیر [۳۰] ماشینی است که می‌تواند تعداد متناهی عمل انجام دهد. هر عملِ انتخاب‌شده توسط یک محیط احتمالی ارزیابی شده و نتیجه‌ی آن در قالب پاسخی مثبت یا منفی به اتوماتا بازخورد می‌شود. اتوماتا از این پاسخ استفاده کرده و خود را بهبود می‌دهد تا اعمال بعدی را بهتر انتخاب کند. بهترین عمل آنست که احتمال دریافت پاداش از محیط را به حداکثر برساند.

[1] Stigmergy
[2] Traveling Salesman Problem
[3] Quadratic Assignment Problem
[4] Job-Shop Scheduling Problem
[5] Vehicle Routing Problem
[6] Graph Sequential Ordering
[7] Communication Network Routing
[8] Earliest Start Time (EST)

هدف نهایی این است که اتوماتا یاد بگیرد همواره عمل مناسبتر را از میان اعمال مجاز خود انتخاب کند. امروزه اتوماتای یادگیر به عنوان یکی از مهمترین ابزارهای محاوره با محیط، کاربردهای فراوانی پیدا کرده است [۳۱]. شکل (۱) چگونگی ارتباط بین اتوماتای یادگیر و محیط را نشان می‌دهد.

شکل (۱): ارتباط بین محیط و اتوماتای یادگیر

بسیاری از مسایل پیچیده را نمی‌توان با استفاده از یک اتوماتای یادگیر تنها حل کرد بلکه قدرت اصلی اتوماتای یادگیر زمانی بهتر آشکار می‌شود که بصورت دسته‌جمعی به کار گرفته شود. بدین ترتیب با ترکیب دو مدل اتوماتای سلولی و اتوماتای یادگیر مدل جدیدی به نام اتوماتای یادگیر سلولی [۳۲] ایجاد می‌شود که نقاط ضعف دو مدل قبلی را ندارد و با موفقیت بر روی مسایل مختلفی چون پردازش تصویر، مدلسازی شبکه‌های تجاری، تخصیص کانال در شبکه‌های سلولی و ... بکار گرفته شده است [۳۳]. در واقع یک اتوماتای یادگیر سلولی، یک اتوماتای سلولی است که هر سلول آن به یک یا چند اتوماتای یادگیر مجهز شده است. در هر لحظه هر اتوماتای یادگیر در اتوماتای یادگیر سلولی با توجه به وضعیت داخلی خود یک عمل از مجموع اعمال مجاز خود را انتخاب می‌کند. عمل انتخاب‌شده با توجه به عمل انتخاب‌شده توسط سلول‌های مجاور و قانون محلی حاکم بر اتوماتای یادگیر سلولی، پاداش یا جریمه می‌شود. اتوماتا از این بازخورد استفاده کرده و توسط یک الگوریتم یادگیری مناسب، ساختار داخلی خود را به‌هنگام می‌کند. در صورتی که عمل بروزرسانی تمام اتوماتاها به صورت همزمان انجام شود آنرا اتوماتای سلولی همگام[9] و در غیر اینصورت ناهمگام[10] گویند. فرآیند انتخاب عمل و دادن پاداش/جریمه تا زمانی که سیستم به حالت پایداری برسد یا معیار از پیش‌تعریف‌شده‌ای برقرار شود ادامه دارد. اتوماتای یادگیر سلولی را یکنواخت گویند اگر تابع همسایگی، قانون محلی و اتوماتای یادگیر تمامی سلول‌ها یکسان باشد و در غیر اینصورت غیریکنواخت نامیده می‌شود. اگر اتصال بین سلول‌ها در یک شبکه منظم مانند آرایه باشد آنرا اتوماتای یادگیر سلولی منظم و در غیر

[9] Synchronous
[10] Asynchronous

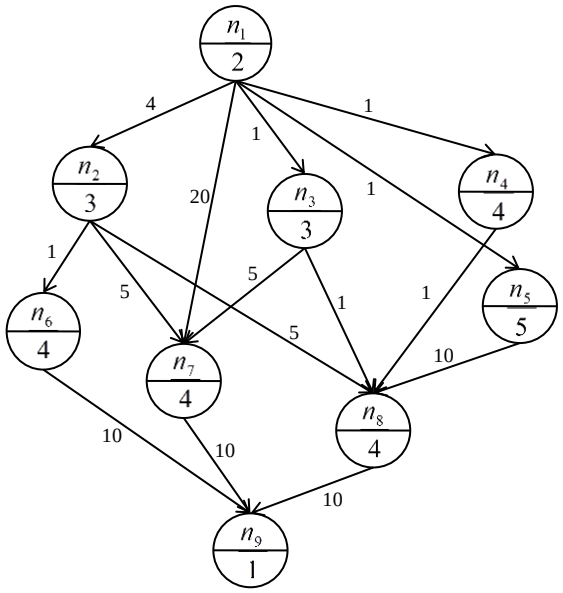
جدول (۱): معیارهای اولویت مختلف برای هر گره در گراف وظایف شکل (۲)

Node	*TLevel*	*BLevel*	*SLevel*	*ALAP*	*NOO*
n_1	0	37	12	0	8
n_2	6	23	8	14	4
n_3	3	23	8	14	3
n_4	3	20	9	17	2
n_5	3	30	10	7	2
n_6	10	15	5	22	1
n_7	22	15	5	22	1
n_8	18	15	5	22	1
n_9	36	1	1	36	0

اینصورت نامنظم نامیده می‌شود. اتوماتای یادگیر سلولی مورد استفاده در این تحقیق یک اتوماتای سلولی یادگیر همگام، یکنواخت و نامنظم است.

۲- زمانبندی کارها در سیستمهای چندپردازنده‌ای

یک گراف جهت‌دار غیرحلقوی $\mathbf{G} = \{\mathbf{N}, \mathbf{E}, \mathbf{W}, \mathbf{C}\}$ بنام گراف وظایف در مدلسازی مساله زمانبندی کارها در سیستم‌های چندپردازنده‌ای بکار گرفته می‌شود که $\mathbf{N} = \{n_1, n_2, \ldots, n_n\}$، $\mathbf{E} = \{(n_i, n_j) \mid n_i, n_j \in \mathbf{N}\}$، $\mathbf{W} = \{w_1, w_2, \ldots, w_n\}$ ، $\mathbf{C} = \{c(n_i, n_j) \mid (n_i, n_j) \in \mathbf{E}\}$ و n به ترتیب مجموعه گره‌ها، مجموعه یال‌ها، مجموعه وزن گره‌ها، مجموعه وزن یال‌ها و تعداد گره‌ها می‌باشند. شکل (۲) گراف وظایف یک برنامه متشکل از ۹ کار را نشان می‌دهد. در هر گراف وظایف، گره‌ها بیانگر کارها و یال‌ها مبین قیود اولویت بین کارها هستند. وجود یال $(n_i, n_j) \in \mathbf{E}$ نشان می‌دهد که کار n_i باید خاتمه یابد تا کار n_j بتواند شروع شود. زیرا کار n_j از داده‌های تولیدشده توسط کار n_i استفاده می‌کند. در چنین شرایطی به گره n_i گره پدر یا والد و به گره n_j گره فرزند گفته می‌شود. گرهی که هیچ گره والدی در گراف نداشته باشد یک گره ورودی[1] و گرهی که هیچ فرزندی نداشته باشد یک گره خروجی[2] است. وزن روی هر گره مانند w_i زمان لازم جهت اجرای کار n_i است و وزن روی هر یال مانند $c(n_i, n_j)$ زمان لازم جهت ارسال داده از کار n_i به کار n_j می‌باشد یعنی پس از اتمام کار n_i، کار n_j باید به اندازه $c(n_i, n_j)$ صبر کند تا داده‌ها آماده گردد و سپس می‌تواند اجرا شود. تنها در شرایطی که هر دو کار روی یک پردازنده یکسان اجرا شوند از این زمان صرفه‌نظر می‌شود چون داده‌ها روی این پردازنده مهیاست و نیاز به انتقال داده به پردازنده دوم از بین می‌رود. در زمانبندی ایستا، زمان لازم جهت اجرای



شکل (۲): گراف وظایف یک برنامه متشکل از ۹ کار [۲۷]

کارها، قیود اولویت و هزینه‌های ارتباطی در حین مرحله کامپایل برنامه محاسبه می‌شوند. کارهای موجود در گراف وظایف باید جهت اجرا به گونه‌ای به m پردازنده فرستاده شوند که قیود اولویت رعایت شده و زمان اتمام کل کارها (یعنی زمان اتمام برنامه) حداقل گردد.

اغلب الگوریتم‌های زمانبندی BNP از تکنیک به نام زمانبندی لیستی استفاده می‌کنند [۳۷]. بدین ترتیب که در هر مرحله لیستی از کارهای آماده جهت اجرا به نام لیست آماده[3] می‌سازند. کارهای آماده جهت اجرا آنهایی هستند که پدر ندارند و یا تمامی اجدادشان قبلاً زمانبندی شده‌اند. سپس به هر گره اولویتی منتسب می‌کنند. معیارهای اولویت مختلف جهت انتساب به کارها در ادامه بررسی خواهند شد. بدین ترتیب پراولویت‌ترین کار از لیست آماده انتخاب شده و به پردازنده‌ای که کمترین زمان شروع ممکن را فراهم کند داده می‌شود. با اجرای این کار، کار مورد نظر از لیست آماده خط خورده و فرزندان آن که امکان اجرا پیدا می‌کنند به این لیست اضافه می‌شوند. این مراحل تا زمانبندی کامل کارها ادامه می‌یابد.

با توجه به وجود قیود اولویت و هزینه‌های ارتباطی بین کارها، اگر تمامی پدران کار n_i روی پردازنده p_j اجرا شده باشند آنگاه کمترین زمان ممکن جهت شروع این کار روی این پردازنده برابر $Avail(p_j)$ است یعنی زودترین زمانی که پردازنده p_j جهت اجرای کار بعدی آزاد می‌شود و در غیر اینصورت باید از طریق فرمول (۱) محاسبه شود.

[1] Entry node
[2] Exit node
[3] Ready-List

$$EST(n_i, p_j) = \max\left(Avail(p_j), \max_{n_m \in Parents(n_i)} \left(FT(n_m) + c(n_m, n_i)\right)\right) \quad (۱)$$

که $FT(n_m) = EST(n_m) + w_m$ برابر زمان واقعی اتمام کار n_m و $Parents(n_i)$ مجموعه تمامی پدران کارِ n_i می باشد. در نهایت زمان اتمام برنامه (کل کارها) با ماکزیمم‌گیری روی زمان اتمام کارها و از طریق فرمول زیر محاسبه می‌شود:

$$makespan = \max_{i=1}^{n}\left(FT(n_i)\right) \quad (۲)$$

پیاده‌سازی‌ها با استفاده از ماتریس مجاورتی نشاندهنده آنند که جهت یافتن $EST(n_i, p_j)$ میباست تمام کارهای موجود در گراف وظایف بررسی شوند تا پدران n_i مشخص شده و در صورت لزوم هزینه‌های ارتباطی بین کار n_i و پدرانش لحاظ گردد؛ لذا پیچیدگی زمانی آن $O(n)$ خواهد بود که همانطور که اشاره شد n تعداد کل کارها در گراف وظایف ورودی است. همچنین جهت یافتن $EST(n_i)$ میبایست کمترین زمان شروع ممکن برای کار n_i بر روی تمام m پردازنده موجود بررسی شود؛ لذا پیچیدگی زمانی آن $O(mn)$ است. بدین ترتیب اجرای الگوریتم انتساب به روش کمترین زمان شروع ممکن برای تمامی کارهای موجود در یک گراف وظایف، پیچیدگی زمانی $O(mn^2)$ خواهد داشت.

معیارهای اولویت مختلفی جهت انتساب به کارها در زمانبندی لیستی گراف وظایف می‌توان استفاده کرد [۳۸] که برخی از آنها عبارتند از سطح بالایی[1]، سطح پایینی[2]، سطح ایستا[3]، دیرترین زمان ممکن[4] و تعداد نوادگان[5]. سطح بالایی یا زودترین زمان ممکن برای گره n_i برابر است با طول بلندترین مسیر از یک گره ورودی تا گره n_i به استثنا خودش. که طول یک مسیر مجموع وزن گره‌ها و یال‌های موجود در آن مسیر است. سطح بالایی هر گره در گراف وظایف را می‌توان با پیمایش توپولوژیکی گراف بشکل زیر محاسبه کرد:

$$TLevel(n_i) = \max_{n_j \in Parents(n_i)} \left(TLevel(n_j) + c(n_j, n_i) + w_j\right) \quad (۳)$$

سطح پایینی هر گره مانند n_i برابر است با طول بلندترین مسیر از گره n_i تا یک گره خروجی که می‌تواند برای هر گره با استفاده از پیمایش توپولوژیکی گراف وظایف بطور معکوس محاسبه شود:

$$BLevel(n_i) = \max_{n_j \in Children(n_i)} \left(BLevel(n_j) + c(n_i, n_j)\right) + w_i \quad (۴)$$

که $Children(n_i)$ مجموعه تمامی فرزندان گره n_i است. در صورتی که در محاسبه طول مسیر برای سطح پایینی هر گره، وزن یال‌ها در نظر گرفته نشود آنگاه معیار اولویت جدید به نام سطح ایستای گره بدست می‌آید:

$$SLevel(n_i) = \max_{n_j \in Children(n_i)} \left(SLevel(n_j)\right) + w_i \quad (۵)$$

دیرترین زمان ممکنِ یک گره معیاری است که نشان می‌دهد آن کار به چه میزان می‌تواند منتظر بماند طوری که طول زمانبندی کلی تغییری نکند. این معیار را می‌توان با استفاده از فرمول زیر بدست آورد:

$$ALAP(n_i) = \min_{n_j \in Children(n_i)} \left(CPL, ALAP(n_j) - c(n_i, n_j)\right) - w_i \quad (۶)$$

که CPL طول مسیر بحرانی (Critical Path) گراف است یعنی طول بلندترین مسیر در گراف وظایف مربوطه.

تعداد نوادگان هر گره مانند n_i معیار بعدی است که به سادگی با پیمایش توپولوژیکی گراف و شمارش نوادگان هر گره محاسبه می‌شود:

$$NOO(n_i) = 1 + NOO(n_j) \quad \forall n_j \in Children(n_i) \quad (۷)$$

البته پس از ملاقات هر گره، آن گره باید علامت‌گذاری شود تا از شمارش مجدد گره‌ها جلوگیری به عمل آید.

جدول (۱) لیستی از معیارهای اولویت ذکرشده را برای هر گره از گراف وظایف شکل (۲) نشان می‌دهد. با استفاده از این معیارها، الگوریتم‌های متنوعی جهت زمانبندی گراف وظایف پیشنهاد شده‌اند که برخی از مهمترین آنها در ادامه بررسی خواهند شد.

۲-۱- الگوریتم ابتدا بالاترین سطح با تخمین زمان[6]

این الگوریتم ابتدا سطح بالایی هر گره را محاسبه می‌کند سپس لیست آماده‌ها را ساخته و در ابتدای هر مرحله به ترتیب نزولی برحسب سطح بالایی مرتب می‌کند. این الگوریتم در هر مرحله، اولین گره از لیست آماده‌ی مرتب‌شده را انتخاب کرده و به پردازنده‌ای که زودترین زمان شروع را مهیا می‌کند زمانبندی می‌کند. در ادامه آن گره را از لیست آماده حذف کرده و فرزندان آنرا که اکنون می‌توانند اجرا شوند به لیست آماده اضافه می‌کند. اینکار تا زمانبندی کل کارها ادامه دارد. شکل (۳)-الف نمودار گانت حاصل از زمانبندی گراف وظایف شکل (۲) را با استفاده از این الگوریتم روی دو پردازنده نشان می‌دهد. در این نمودار فضای خالی بین n_3 و n_8 بیانگر اینست که هر چند که n_8 می‌توانست پس از n_3 اجرا شود اما از آنجا که دو تا از پدارانش یعنی n_2 و n_5 روی پردازنده دیگری اجرا شده‌اند لذا پس از اتمام آنها باید به اندازه بزرگترین هزینه ارتباطی یعنی ۱۰ واحد زمانی صبر کند تا داده‌ها روی پردازنده p_2 مهیا شده و سپس اجرا شود.

[1] Top-Level
[2] Bottom-Level
[3] Static-Level
[4] As-Late-As-Possible
[5] Number-Of-Offspring
[6] HLFET

در صورت پیاده‌سازی گراف وظایف با استفاده از ماتریس مجاورتی، در زیرمساله ترتیب کارها، بیشترین هزینه زمانی این الگوریتم صرف پیمایش ماتریس و یافتن اولویت سطح بالایی هر گره و همچنین مرتب‌سازی کارها برحسب این معیار می‌شود که هر کدام دارای پیچیدگی زمانی $O(n^2)$ می‌باشند. پس از ایجاد ترتیب مناسب اجرای کارها، الگوریتم انتساب زودترین زمان شروع ممکن با پیچیدگی زمانی $O(mn^2)$ عملیات انتساب کارها به پردازنده‌ها را انجام خواهد داد.

۲-۲- الگوریتم مسیر بحرانی تعدیل‌شده[1]

این الگوریتم از معیار دیرترین زمان ممکنِ یک گره به عنوان اولویت زمانبندی استفاده می‌کند. بدین ترتیب که در ابتدا دیرترین زمان ممکنِ هر کار را محاسبه کرده و لیست آماده را به ترتیب صعودی بر حسب آن مرتب می‌کند. تساوی‌ها با استفاده از دیرترین زمان اجرای ممکن فرزندان کارها شکسته می‌شود. سپس در هر مرحله، اولین گره از لیست آماده‌ی مرتب‌شده را انتخاب کرده و به پردازنده‌ای که زودترین زمان شروع را مهیا می‌کند بصورت درجی زمانبندی می‌کند. در ادامه آن گره را از لیست آماده حذف کرده و فرزندان آنرا که اکنون می‌توانند اجرا شوند به لیست آماده اضافه می‌کند. اینکار تا زمانبندی کل کارها ادامه دارد. شکل (۳)-ب نمودار گانت حاصل از زمانبندی گراف وظایف شکل (۲) را با استفاده از این الگوریتم روی دو پردازنده نشان می‌دهد.

در صورت پیاده‌سازی گراف وظایف با استفاده از ماتریس مجاورتی، در زیرمساله ترتیب کارها، بیشترین هزینه زمانی این الگوریتم صرف مرتب‌سازی کارها برحسب معیار اولویت دیرترین زمان ممکنِ هر کار می‌شود که می‌بایست تساوی‌ها نیز با استفاده از دیرترین زمان اجرای ممکن فرزندان کارها شکسته شود؛ لذا پیاده‌سازی این الگوریتم با پیچیدگی زمانی $O(n^3)$ امکانپذیر خواهد بود. پس از ایجاد ترتیب مناسب اجرای کارها، الگوریتم انتساب زودترین زمان شروع ممکن بصورت درجی با پیچیدگی زمانی $O(mn^2)$ عملیات انتساب کارها به پردازنده‌ها را انجام خواهد داد. در روش انتساب درجی، پس از مشخص‌شدن پردازنده‌ای که کمترین زمان شروع ممکن را فراهم می‌کند و در صورتی که انتساب کار موردنظر به آن پردازنده حفره‌ای را در زمانبندی ایجاد کند؛ الگوریتم سعی می‌کند تا حفره بوجودآمده را با جاسازی مناسب‌ترین کارهای موجود در لیست آماده پر کند به این امید که طول زمانبندی نهایی کاهش یابد. هر چند انتساب کارها به هر دو روش معمولی و درجی از پیچیدگی زمانی $O(mn^2)$ برخوردار هستند؛ با این وجود روش درجی سربار محاسباتی سنگینی به سیستم وارد می‌کند.

۲-۳- الگوریتم زمانبندی سطح پویا[2]

این الگوریتم از یک معیار اولویت به نام سطح پویا استفاده می‌کند که تفاضل بین سطح ایستای گره و زودترین زمان شروع ممکن آن گره است. در هر مرحله، سطح پویا برای تمامی گره‌های موجود در لیست آماده و روی تمامی پردازنده‌ها محاسبه می‌شود. سپس زوج گره و پردازنده‌ای که بزرگترین مقدار سطح پویا را ایجاد می‌کنند جهت زمانبندی انتخاب می‌شوند. این الگوریتم در ابتدا تمایل به زمانبندی گره‌ها به ترتیب نزولی سطح ایستا و در انتهای زمانبندی تمایل به ترتیب صعودی سطح بالایی گره‌ها دارد. . شکل (۳)-ج نمودار گانت حاصل از زمانبندی گراف وظایف شکل (۲) را با استفاده از این الگوریتم روی دو پردازنده نشان می‌دهد.

در خصوص محاسبه پیچیدگی زمانی این الگوریتم لازم به ذکر است که حلقه اصلی الگوریتم مذکور می‌بایست به تعداد n بار تکرار شود. در هر تکرار باید برای هر کار موجود در لیست آماده مانند n_i مقدار $EST(n_i)$ آن محاسبه شده و با در نظر گرفتن بزرگترین مقدار سطح پویای گره‌ها، زوج کار-پردازنده بعدی جهت زمانبندی و انتساب مستقیم انتخاب شود؛ لذا مرتبه زمانی کل الگوریتم $O(mn^3)$ خواهد بود.

۲-۴- الگوریتم زودترین زمان اجرا اول[3]

این الگوریتم در هر مرحله، زودترین زمان شروع ممکن را برای هر کار موجود در لیست آماده روی تمامی پردازنده‌ها محاسبه می‌کند. سپس از میان آنها زوج گره-پردازنده‌ای که زودترین زمان اجرای ممکن را مهیا می‌کنند انتخاب کرده و زمانبندی می‌کند. در ادامه مطابق سایر الگوریتم‌های این دسته، آن گره را از لیست آماده حذف کرده و فرزندان آنرا که اکنون می‌توانند اجرا شوند به لیست آماده اضافه می‌کند. اینکار تا زمانبندی کل کارها ادامه دارد. شکل (۳)-د نمودار گانت حاصل از زمانبندی گراف وظایف شکل (۲) را با استفاده از این الگوریتم روی دو پردازنده نشان می‌دهد. با توضیحات داده‌شده در مورد الگوریتم زمانبندی سطح پویا، مرتبه زمانی کلی این الگوریتم جهت زمانبندی و انتساب مستقیم نیز برابر $O(mn^3)$ خواهد بود.

[1] MCP

[2] DLS

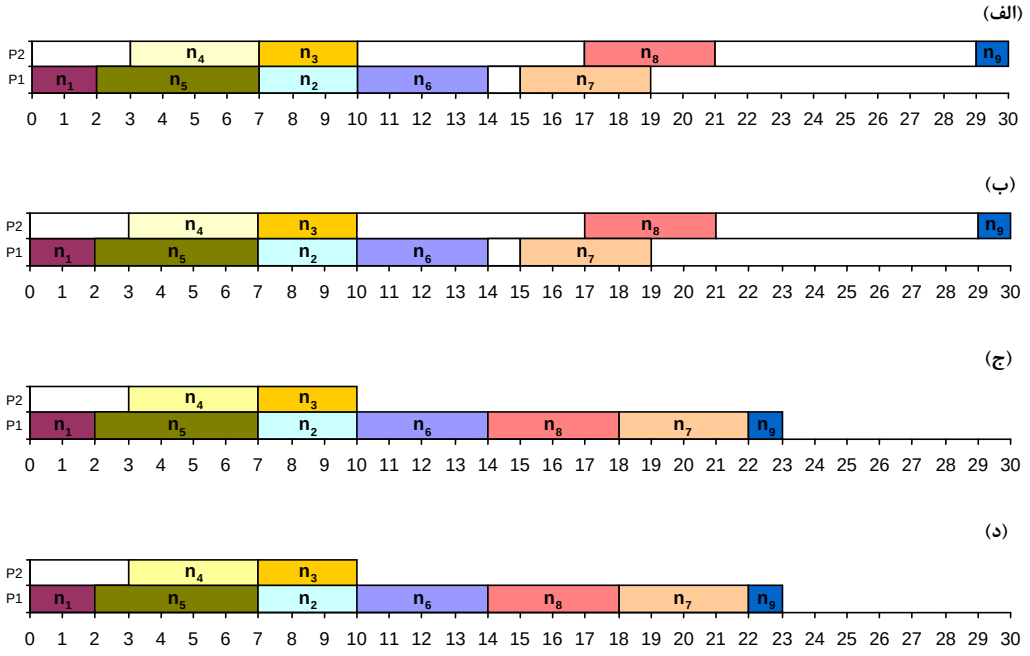
[3] ETF

۳- الگوریتم بهینه‌سازی کلونی مورچه‌ها

با خروج از خانه در ابتدا مورچه‌ها حرکاتی کاملاً تصادفی دارند. به محض آنکه مورچه‌ای یک منبع غذا پیدا کند مقداری از آنرا برداشته و در مسیر بازگشت به سمت لانه شروع به ترشح ماده شیمیایی بوداری به نام فرومون[1] می‌کند. از آنجا که مورچه‌ها مجذوب فرومون می‌شوند. آنها نیز کم‌کم منبع غذا را یافته و در مسیر بازگشت فرومون بیشتری ترشح می‌شود و بدین ترتیب یک مسیر فرومونی از منبع غذایی به لانه تشکیل شده و مورچه‌ها در این مسیر تردد خواهند کرد. فرومون بیشتر باعث جذب مورچه بیشتر شده و مورچه بیشتر فرومون بیشتری ترشح می‌کند به این نوع رفتار، خودکاتالیزوری[2] گویند. بدین ترتیب مورچه‌ها با استفاده از مسیرهای فرومونی دارای ارتباط محلی غیرمستقیمی هستند که بدان استیگمرجی گویند. استیگمرجی باعث می‌شود در حرکات جدید مورچه‌ها، تمامی تجربیات مورچه‌های پیشین در نظر گرفته شود و از جمله عواملی است که این الگوریتم را بسیار موثر و سریع می‌کند. فرومون موجود در محیط، هر لحظه توسط گرما و نور خورشید در حال تبخیر است و مسیرهای هرز و اضافی که منبع غذایشان به پایان رسیده را محو می‌کند تا حرکات تصادفی مورچه‌ها در یافتن منابع غذایی دیگر مجدداً آغاز شود.

اگر مانعی بر سر راه مورچه‌ها قرار گیرد و مسیر فرومونی را قطع کند ابتدا مورچه‌ها بطور کاملاً تصادفی از دو طرف مانع را دور زده تا به مسیر فرومونی خود بازگردند. اما از آنجا که مورچه‌ها طرف بلندتر، دیرتر می‌رسند و فرومون نیز هر لحظه در حال تبخیر است لذا فرومون کمتری در طرف بلندتر باقی می‌ماند و مورچه کمتری به طرف بلندتر جذب می‌شود و کم‌کم اکثر مورچه‌ها جذب طرف کوتاه‌تر شده و بدین ترتیب مورچه‌ها همواره مسیرهای کوتاهتر را می‌یابند.

الگوریتم بهینه‌سازی کلونی مورچه‌ها سعی در شبیه‌سازی این رفتار کاوشگرانه دارد. بدین ترتیب که در ابتدا به هر حالت از مساله یک متغیر عددی به نام اثر فرومونی منتسب می‌شود. مقادیر اولیه این متغیرها همگی یکسان و برابر مقدار ناچیزی است. الگوریتم بهینه‌سازی کلونی مورچه‌ها یک الگوریتم تکرارشونده است که در هر تکرار یک یا چند مورچه ایجاد می‌شود. در حقیقت هر مورچه مصنوعی تنها یک لیست است که حالت‌های ملاقات‌شده توسط آن مورچه را نگه‌داری می‌کند. مورچه ایجادشده بر روی حالت ابتدایی قرار داده می‌شود یا به عبارت دیگر حالت ابتدایی درون لیست مورچه قرار می‌گیرد. مورچه حالت بعدی را با استفاده از یک تصمیم‌گیری احتمالی بر اساس میزان فرومون حالت‌های مجاور و مطلوبیت آنها از نظر مساله برمی‌گزینند و این حالت را به لیست خود اضافه می‌کند. مورچه این کار را تا رسیدن به یک حالت نهایی ادامه می‌دهد. در این زمان، میزان مطلوبیت راه‌حل بدست‌آمده توسط مورچه محاسبه شده و بر حسب آن، میزان فرومون حالت‌های ملاقات‌شده افزایش می‌یابد. در نهایت



شکل (۳): زمانبندی گراف وظایف شکل (۲) با استفاده از چهار رهیافت اکتشافی سنتی معرفی‌شده: الف) HLFET ب) MCP ج) DLS د) ETF

[1] Pheromone
[2] Autocatalysis

تمامی مقادیر فرومونی بطور یکسانی کاهش می یابند تا تبخیر فرومون در محیط واقعی شبیه‌سازی شود. تبخیر فرومون بسیار مهم بوده و از گیرافتادن مورچه‌ها در حداقل‌های محلی و همگرایی زودرس جلوگیری می‌کند.

با تکرار این الگوریتم، کم‌کم مورچه‌ها به سمت مسیرهای بهینه‌تر همگرا می‌شوند. یکی از برتری‌های الگوریتم کلونی مورچه‌ها نسبت به رهیافت‌های تکاملی چون الگوریتم ژنتیک، ارتباط غیرمستقیم محلی مورچه‌ها با استفاده از اثرات فرومونی است (استیگمرجی). جایی که برخلاف الگوریتم ژنتیک که تمامی تصمیمات تصادفی بوده و بر اساس ضریب تبادل و جهش است و بسیاری از تجربیات نیز در حین مرحله انتخاب از بین می‌روند در الگوریتم بهینه‌سازی کلونی مورچه‌ها تمامی تصمیمات هدفمند و بر اساس تجربیات بدست‌آمده از مورچه‌های قبلی است. بدین ترتیب این الگوریتم معمولاً جواب‌های بهتر و سریعتری نسبت به الگوریتم‌های تکاملی چون ژنتیک می‌یابد.

۴- اتوماتای یادگیر سلولی

اتوماتای یادگیر یک مدل انتزاعی جهت محاوره با محیط احتمالی است که می‌تواند مجموعه محدودی از عملیات را روی محیط انجام دهد. انتخاب یک عمل در هر لحظه بسته به وضعیت درونی اتوماتا در آن لحظه دارد که با استفاده از یک بردار احتمال عمل نمایش داده می‌شود. محیط احتمالی به عمل انتخاب‌شده توسط اتوماتا واکنش نشان داده و آنرا ارزیابی می‌کند. اتوماتا با استفاده از بازخورد عملِ انتخاب‌شده، خود را پالایش کرده و بردار احتمال عمل‌هایش را به روز می‌کند. بهترین عمل آنست که از طرف محیط بیشترین پاداش و کمترین جریمه را به همراه داشته باشد. محیط احتمالی را با استفاده از سه‌تایی $\mathbf{E} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{c}\}$ نشان می‌دهند که $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ مجموعه ورودی‌های محیط، $\boldsymbol{\beta} = \{\beta_1, \beta_2, \ldots, \beta_m\}$ مجموعه واکنش‌های محیط و $\mathbf{c} = \{c_1, c_2, \ldots, c_r\}$ مجموعه احتمال‌های جریمه هر عمل ورودی توسط محیط بوده و r و m به ترتیب تعداد ورودیها و خروجی‌های مجاز محیط می‌باشند. مجموعه $\boldsymbol{\beta}$ می‌تواند دو عضوی باشد که در اینصورت $\beta_1 = 0$ بیانگر پاداش و $\beta_2 = 1$ نشاندهنده جریمه از طرف محیط است. مقادیر $\mathbf{c}$ در محیط‌های ایستا ثابت است حال آنکه در محیط‌های پویا ممکن است تغییر کنند. در حالت دوم اتوماتای یادگیر می‌تواند با ساختار متغیر تعریف شده و خود را با استفاده از یک الگوریتم یادگیری محاوره‌ای با محیط وفق دهد.

بطور کلی اتوماتای یادگیر با ساختار متغیر را با یک چهارتایی $\mathbf{LA} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{p}, T\}$ نشان می‌دهند که $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$, $\boldsymbol{\beta} = \{\beta_1, \beta_2, \ldots, \beta_m\}$, $\mathbf{p} = \{p_1, p_2, \ldots, p_r\}$, T, r, and m به ترتیب مجموعه عمل‌هایی که می‌تواند توسط اتوماتا انتخاب شود، مجموعه ورودی‌های اتوماتا (انواع پاسخ‌های احتمالی محیط)، بردار احتمال هر عمل، الگوریتم یادگیری، تعداد عملیات مجاز قابل انتخاب توسط اتوماتا و تعداد ورودی‌های اتوماتاست.

الگوریتم یادگیری از بازخورد ایجادشده توسط محیط استفاده کرده و بردار احتمال عمل اتوماتا را به گونه‌ای مناسب طوری تغییر می‌دهد که عمل‌های بعدی برای محیط مطلوب‌تر باشند. هر چند که الگوریتم‌های یادگیری متنوعی برای اتوماتای یادگیر پیشنهاد شده‌اند در ادامه یکی از موثرترین آنها به نام الگوریتم یادگیری خطی بررسی می‌شود. فرض کنید $\alpha_i(n)$ عملی باشد که در مرحله n توسط اتوماتای LA^k انتخاب شده است و احتمال انتخاب این عمل $p_i(n)$ باشد. این عمل به محیط اعمال شده و واکنش محیط گرفته می‌شود. اگر پاسخ محیط مطلوب بود آنگاه بردار احتمال عمل آن اتوماتا $\boldsymbol{p}^k(n)$ با استفاده از فرمول (۸) به‌روزرسانی شده و عمل انتخاب‌شده را تشویق می‌کند.

$$p_j^k(n+1) = \begin{cases} p_j^k(n) + a \times \left(1 - p_j^k(n)\right) & if \quad j = i \\ (1-a) \times p_j^k & if \quad j \neq i \end{cases} \qquad (۸)$$

از طرف دیگر هنگامی که پاسخ محیط نامطلوب باشد آنگاه عمل انتخاب‌شده توسط اتوماتا با فرمول (۹) جریمه می‌شود.

$$p_j^k(n+1) = \begin{cases} p_j^k(n) \times (1-b) & if \quad j = i \\ \dfrac{b}{r-1} + (1-b) \times p_j^k(n) & if \quad j \neq i \end{cases} \qquad (۹)$$

که a و b به ترتیب ضریب پاداش و ضریب جریمه هستند که به طور تجربی تنظیم می‌شوند. اگر $a = b$ انتخاب شود الگوریتم را L_{R-P} در غیر اینصورت اگر a بسیار بزرگتر از b انتخاب شود الگوریتم $L_{R-\varepsilon P}$ و در شرایطی که $b = 0$ باشد الگوریتم L_{R-I} است.

اتوماتای سلولی d-بعدی یک چندتایی بشکل $\mathbf{CA} = \{Z^d, \boldsymbol{\phi}, N, F\}$ است که Z^d یک شبکه منظم از d-تایی‌های مرتب از سلول‌هاست، $\boldsymbol{\phi} = \{1, \ldots, m\}$ یک مجموعه متناهی از حالت‌هاست که هر سلول در هر لحظه می‌تواند در یکی از این حالت‌ها باشد و m تعداد حالت‌های مجاز است. $\bar{x} \in Z^d \mid N = \{\bar{x}_1, \ldots, \bar{x}_{\bar{m}}\}$ بردار همسایگی است که موقعیت نسبی همسایگان هر سلول در اتوماتای سلولی را مشخص می‌کند و $F : \phi_{\bar{m}} \rightarrow \phi$ قانون محلی حاکم بر **CA** است که روی بردار همسایگی آن تعریف شده و با توجه به حالت‌های جاری همسایگان، اتوماتای سلولی را از حالتی به حالت دیگر می‌برد.

در مدلسازی سیستم‌های واقعی گاهی لازم است که قوانین بصورت غیرقطعی و احتمالی در نظر گرفته شوند چیزی که اتوماتای سلولی از آن عاجز است. همچنین در محیط‌های ناشناخته استخراج مستقیم قوانین و احتمال رخدادها عملاً غیرممکن است.

اتوماتای یادگیر بهترین ابزار جهت محاوره با اینگونه محیط‌هاست. بدین ترتیب، با ترکیب دو مدل اتوماتای سلولی و اتوماتای یادگیر، مدل جدیدی به نام اتوماتای یادگیر سلولی ایجاد می‌شود که فاقد نقاط ضعف دو مدل مذکور است. یک اتوماتای یادگیر سلولی d-بعدی یک چندتایی به شکل $\mathbf{CLA} = \{Z^d, \phi, A, N, F, T\}$ است که Z^d یک شبکه از d-تایی‌های مرتب از سلول‌هاست (در صورتی که اتوماتا منظم باشد)، $\phi = \{1, \ldots, m\}$ یک مجموعه متناهی از حالت‌هاست که هر سلول در هر لحظه می‌تواند در یکی از این حالت‌ها باشد و m تعداد حالت‌های مجاز است. A همان اتوماتای یادگیر موجود در هر سلول از اتوماتای یادگیر سلولی است که می‌تواند بصورت یگانه یا چندگانه باشد. $\bar{x} \in Z^d \mid N = \{\bar{x}_1, \ldots, \bar{x}_{\bar{m}}\}$ یک بردار همسایگی است که موقعیت نسبی همسایگان هر سلول در اتوماتای سلولی را مشخص می‌کند و $F : \phi_{\bar{m}} \to \beta$ قانون محلی حاکم بر اتوماتای یادگیر سلولی است که روی بردار همسایگی آن سلول تعریف شده و با توجه به حالت‌های جاری همسایگان، سیگنال تقویتی β را به عنوان بازخورد از محیط جهت آموزش اتوماتای یادگیر درون سلول تولید می‌کند. مجدداً مجموعه $\boldsymbol{\beta}$ می‌تواند دو عضوی باشد که در اینصورت $\beta_1 = 0$ بیانگر پاداش و $\beta_2 = 1$ نشاندهنده جریمه از طرف محیط است و در نهایت T الگوریتم یادگیری است که می‌تواند مطابق فرمول‌های (۸) و (۹) اعمال شود.

اتوماتای یادگیر سلولی نامنظم یک اتوماتای یادگیر سلولی است که دارای ساختار شبکه‌ای منظم مانند آرایه یا ماتریس نمی‌باشد. یعنی Z^d بشکل یک گراف، درخت و امثالهم تعریف می‌گردد. این نوع از اتوماتا در پیاده‌سازی مسایلی مانند شبکه حسگر، سیستم‌های شبکه‌ای آزاد و کاربردهای مبتنی بر گراف که در یک شبکه مستطیلی منظم نمی‌توانند مدل شوند استفاده می‌شود. اتوماتای یادگیر سلولی نامنظم بشکل یک گراف تعریف می‌شود که هر گره آن معادل یک سلول است که مجهز به یک اتوماتای یادگیر می‌باشد. بسته به کاربرد انواع همسایگی روی این گراف قابل تعریف است. درست شبیه اتوماتای یادگیر سلولی، قانونی وجود دارد که اتوماتا تحت آن عمل می‌کند و با بروزرسانی اتوماتای یادگیر درون سلول‌ها با توجه به الگوریتم یادگیری، اتوماتون‌ها به‌هنگام شده و خود را محیط وفق می‌دهند [۳۴].

۵- رهیافت پیشنهادی

رهیافت پیشنهادی از دو بخش تشکیل شده است که در ادامه مقاله به تشریح و تفسیر آنها خواهیم پرداخت. بخش اول رهیافت، به حل مساله ترتیب کارها با استفاده از الگوریتم بهینه‌سازی کلونی مورچه‌ها می‌پردازد و بخش دوم، وظیفه انتساب بهینه کارهای استخراج‌شده از بخش اول را به پردازنده‌های موجود داراست که از اتوماتای یادگیر و اتوماتای یادگیر سلولی استفاده شده است.

۵-۱- بخش اول: حل مساله ترتیب کارها با استفاده از الگوریتم بهینه‌سازی کلونی مورچه‌ها

بخش اول رهیافت، ترتیب مناسب اجرای کارها در گراف وظایف را مشخص می‌کند که مبتنی بر الگوریتم بهینه‌سازی کلونی مورچه‌هاست. شکل (۴) این الگوریتم را بصورت فلوچارت نشان می‌دهد. در ابتدا ماتریس $n \times n$ به نام τ به عنوان متغیرهای فرمونی حالت‌های مساله در نظر گرفته می‌شود که n برابر تعداد کارها در گراف وظایف ورودی است. در حقیقت τ_{ij} میزان فرومون یال (n_i, n_j) است که درجه مطلوبیت انتخاب کار n_j درست پس از انتخاب کار n_i را مشخص می‌کند. در ابتدا تمامی درایه‌های این ماتریس مقدار یکسان و ناچیزی دارند تا انتخاب‌های اولیه مورچه‌ها بیشتر تصادفی و بر اساس معیار اولویتِ درنظرگرفته‌شده انجام شود. سپس لیستی (یک آرایه تک‌بعدی) به نام Tabu-List به عنوان مورچه ساخته می‌شود تا گره‌های ملاقات‌شده توسط مورچه را نگه‌داری کند. در قدم بعدی لیستی از کارهای آماده جهت اجرا به نام Ready-List ایجاد می‌شود. کارهای آماده آنهایی هستند که پدری ندارند و یا تمامی پدرانشان قبلاً انتخاب و زمانبندی شده‌اند.

سپس برای هر مورچه حلقه‌ای اجرا می‌شود که در هر تکرار، آن مورچه کار بعدی را بر اساس میزان مطلوبیت کارهای موجود در لیست آماده و با استفاه از یک تصمیم‌گیری احتمالی انتخاب می‌کند. میزان مطلوبیت انتخاب هر کار مانند n_j درست پس از انتخاب کار n_i در تکرار t-ام با استفاده از فرمول زیر محاسبه می‌شود:

$$a_{ij}(t) = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_j]^{\beta}}{\sum_{l \in \mathrm{N}(t)}[\tau_{il}(t)]^{\alpha}[\eta_l]^{\beta}} \qquad \forall j \in \mathrm{N}(t) \qquad (۱۰)$$

که $\tau_{ij}(t)$ میزان اثر فرومونی روی یال (n_i, n_j) در تکرار t-ام، η_j مقدار اکتشافی (میزان اولویت) کار n_j و $\mathrm{N}(t)$ مجموعه کارهای آماده‌ی کنونی، α و β پارامترهایی هستند که وزن نسبی مقادیر فرومونی و میزان اولویت گره‌ها را کنترل می‌کنند. اگر $\alpha > \beta$ باشد آنگاه مقادیر فرومونی اهمیت بیشتری پیدا می‌کنند و اگر $\alpha < \beta$ باشد آنگاه معیار اولویت کارها در تصمیم‌گیری‌ها موثرتر می‌شود. از معیارهای اولویت مختلفی مانند *TLevel*، *BLevel*، *SLevel*، *ALAP* و *NOO* می‌توان به عنوان مقادیر اکتشافی η_j استفاده کرد که بهترین آنها باید بصورت آماری و استنباطی و از طریق اجرای آزمایشات مربوطه استخراج شود. سپس احتمال

انتخاب کار n_j درست پس از انتخاب کار n_i توسط مورچه k-ام در تکرار t-ام بوسیله فرمول زیر بدست می‌آید:

$$p_{ij}^{k}(t)=\frac{a_{ij}(t)}{\sum_{l\in N(t)}a_{il}(t)} \qquad (۱۱)$$

با در اختیار داشتن احتمال انتخاب کارهای موجود در لیست آماده، ابتدا یک عدد تصادفی در بازه [0, 1) تولید شده و بر اساس آن، کار بعدی بشکل چرخ رولت[1] انتخاب می‌شود. بدیهی است کارهایی از لیست آماده که مقادیر فرومونی بیشتر و معیار اولویت بالاتری دارند شانس بهتری جهت انتخاب پیدا می‌کنند. در نهایت، کار انتخاب‌شده به لیست مورچه اضافه شده، از لیست آماده حذف شده و فرزندان آن که اکنون قابل اجرا هستند به لیست آماده اضافه می‌شوند. این عملیات تا پرشدن لیست مورچه که معادل زمانبندی کامل گراف وظایف است ادامه دارد.

پس از آنکه ترتیب مناسب کارها جهت اجرا بدست آمد آنگاه، کارها یکی‌یکی از لیست مورچه خارج شده و به پردازنده‌ها منتسب می‌شوند. در ابتدا از روش انتساب کار به پردازنده‌ای که زودترین زمان شروع ممکن را فراهم می‌آورد استفاده می‌شود تا عملکرد این رهیافت با سایر روش‌های لیستی موجود مقایسه شود. سپس جهت انتساب کارها به پردازنده‌ها روشی مبتنی بر اتوماتای یادگیر سلولی ارائه می‌شود که می‌تواند انتساب‌های بهتری انجام دهد.

با انتساب کارها به پردازنده‌ها، زمانبندی کامل شده و زمان اتمام کل کارها محاسبه می‌شود. با توجه به مطلوبیت زمانبندی بدست‌آمده، میزان فرومون جهت ترشح روی یال‌های ملاقات‌شده توسط مورچه k-ام با فرمول زیر محاسبه می‌گردد.

$$\Delta\tau_{ij}^{k}=\frac{1}{L^{k}} \qquad if\ (n_i,n_j)\in T^{k} \qquad (۱۲)$$

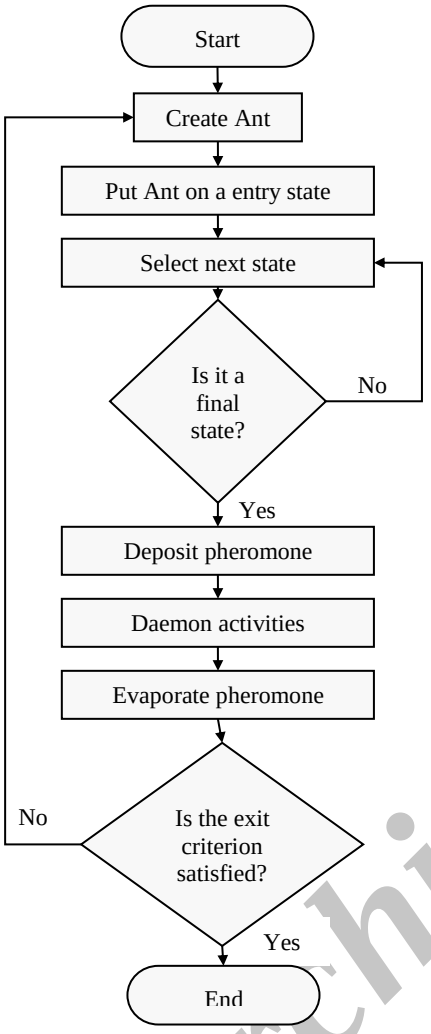
که L^k زمان اتمام بدست‌آمده از مورچه k-ام و T^k تور اجرا شده توسط این مورچه است. یعنی اگر کار n_j درست پس از کار n_i در لیست مورچه باشد آنگاه به اندازه $\Delta\tau_{ij}^{k}$ به τ_{ij} اضافه می‌شود و در غیر اینصورت τ_{ij} دست‌نخورده باقی می‌ماند.

در مرحله بعدی[2] با استفاده از فرمول زیر، مقداری فرمون اضافی بر روی یال‌های ملاقات‌شده توسط بهترین مورچه[3] ترشح می‌شود تا مسیرهای مطلوب بدست‌آمده فراموش نگردند.

$$\Delta\tau_{ij}^{\min}=\frac{1}{L^{\min}} \qquad if\ (n_i,n_j)\in T^{\min} \qquad (۱۳)$$

که L^{min} زمانِ اتمام بدست‌آمده از بهترین مورچه و T^{min} تور اجرا شده توسط این مورچه است. در نهایت با استفاده از فرمول (۱۴) مقادیر فرومونی تمامی درایه‌های ماتریس τ در عدد ρ به نام ضریب تبخیر که در بازه [0, 1] قرار دارد ضرب شده و بدین ترتیب مقادیر فرومونی کاهش می‌یابند که معادل تبخیر فرومون در محیط‌های واقعی است.

$$\tau_{ij}=(1-\rho)\tau_{ij} \qquad (۱۴)$$



شکل(۴): فلوچارت قسمت اول رهیافت پیشنهادی

۵-۲- بخش دوم: حل مساله انتساب کارها به پردازنده‌ها با استفاده از اتوماتای یادگیر سلولی

در روش زمانبندی لیستی پس از بدست‌آمدن ترتیب مناسب جهت اجرا معمولاً الگوریتم‌ها با الهام از روش حریصانه، هر کار را به پردازنده‌ای که زودترین زمان اجرای ممکن را مهیا می‌سازد منتسب می‌کنند. در این روش هر چند هر کار در سریع‌ترین زمان ممکن خود شروع می‌شود اما لزوماً زمان اتمام کل کارها را کمینه نمی‌سازد؛ زیرا علاوه بر زمان شروع هر کار، مناسب‌بودن پردازنده نیز موثر است. یک کار را می‌توان با کمی تاخیر چنان به پردازنده‌یِ مناسب انتساب داد که کارهای بعدی متاثر از آن بتوانند سریعتر آغاز شده و زمان اتمام کل کارها کمتر گردد. با توجه به

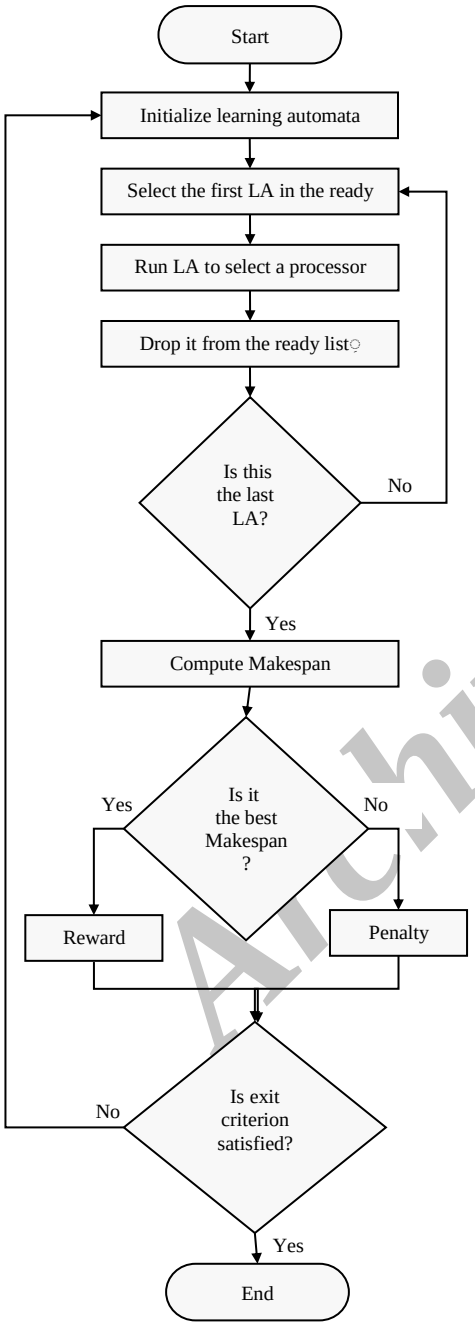
[1] Roulette-wheel Selection
[2] Daemon Activities
[3] Ant^{min}

آنکه هیچ معیاری جهت مشخص‌کردن میزان مطلوبیت یک پردازنده برای اجرای یک کارِ مشخص وجود ندارد و در حقیقت با محیط ناشناخته مواجه هستیم لذا استفاده از اتوماتای یادگیر سلولی در حل این مساله، ایده‌ای بسیار منطقی است.

ابتدا رهیافتی مبتنی بر اتوماتای یادگیر جهت حل این مساله یعنی انتساب کارها به پردازنده‌ها ارائه کرده و با انجام آزمایشاتی نقاط ضعف و قوت آنرا بررسی می‌کنیم. سپس رهیافت پیشنهادی را توسعه داده و رهیافت جدیدی مبتنی بر اتوماتای یادگیر سلولی نامنظم به گونه‌ای ارائه می‌کنیم تا نقاط ضعف رهیافت اول برطرف گردد.

در الگوریتم پیشنهادی اول، در ابتدا به هر کار از گراف وظایف یک اتوماتای یادگیر الصاق می‌گردد. سپس یک الگوریتم تکرارشونده آغاز می‌شود که در هر تکرار، کارها به ترتیبی که جهت اجرا انتخاب شده‌اند یکی‌یکی فعال می‌شوند. هر اتوماتایی که فعال شد یک عدد تصادفی تولید کرده و با توجه به بردار احتمال عمل خود $\mathbf{p} = \{p_1,\ldots,p_r\}$، یکی از پردازنده‌ها را جهت اجرای آن کار انتخاب می‌کند. یعنی هر اتوماتا می‌تواند از بین مجموعه اعمال مجاز خود $\boldsymbol{\alpha} = \{\alpha_1,\ldots,\alpha_r\}$ یکی را انتخاب کند. که r برابر تعداد پردازنده‌های موجود است. پس از آنکه تمامی اتوماتاها پردازنده‌های خود را انتخاب کردند زمانبندی کامل است. حال با ماکزیمم‌گیری روی پردازنده‌ها، زمان اتمام کل کارها محاسبه می‌شود. اگر زمان بدست‌آمده از زمان‌های قبلی کمتر یا مساوی باشد اتوماتاها بطور همگام تشویق می‌شوند و در غیر اینصورت مجازات می‌گردند. در حقیقت ورودی هر اتوماتا (بازخورد از محیط) $\boldsymbol{\beta} = \{0, 1\}$ یک مجموعه دو عضوی است که $\beta_1 = 0$ بیانگر پاداش از محیط و $\beta_2 = 1$ نشاندهنده تنبیه توسط محیط است. هر اتوماتا با توجه به این بازخورد و با استفاده از الگوریتم یادگیری خطی فرمول‌های (۸) و (۹)، خود را بهنگام می‌کند. بدین ترتیب، تکرار جاری تمام شده و تکرار بعدی آغاز می‌شود و اتوماتاها با به هنگام‌سازی خود در هر تکرار، کم‌کم به سمت زمانبندی بهینه همگرا می‌شوند. فلوچارتی از این عملیات در شکل (۵) نشان داده شده است.

الگوریتم دوم که مبتنی بر اتوماتای یادگیر سلولی است دقیقاً مانند رهیافت قبلی به هر کار در گراف وظایف یک سلول الصاق می‌شود که مجهز به یک اتوماتای یادگیر است. همسایگان هر سلول، پدران آن سلول در نظر گرفته می‌شوند. علاوه بر تشویق و مجازاتی که مبتنی بر زمان اتمام کل کارهاست یک قانون محلی نیز روی اتوماتای یادگیر سلولی اعمال می‌شود بدین صورت که اگر یک سلول پردازنده‌ای را جهت اجرا انتخاب کند که تمامی همسایگان آن سلول (تمام پدران آن کار) انتخاب کرده‌اند آنگاه آن سلول پاداش می‌گیرد و در غیر اینصورت تنبیه می‌گردد. این قانون محلی از آنجا نشات می‌گیرد که اگر پدران یک کار همگی روی یک پردازنده اجرا شوند آنگاه هزینه‌های ارتباطی بین پدران و این کار از بین رفته و این کار می‌تواند به محض آزادشدن روی این پردازنده اجرا شود. اما اگر روی پردازنده‌ی دیگری اجرا شود که پدرانش روی آن اجرا نشده‌اند باید به اندازه هزینه‌های ارتباطی صبر کند و بدین ترتیب طول زمانبندی نهایی افزایش خواهد یافت.



شکل (۵): فلوچارت قسمت دوم رهیافت پیشنهادی

۶- جزئیات پیاده‌سازی و نتایج آن

۶-۱- مقدمه‌ای بر سیستم پیاده‌سازی‌شده و مقادیر اولیه

رهیافت پیشنهادی روی یک کامپیوتر رومیزی Pentium4 با پردازنده‌ی هشت‌هسته‌ای آیوی‌بریدج i7-3770K@3.9GHz با 4GB حافظه اصلی DDR III@2133MHz-CL7 روی سیستم عامل Microsoft Windows 7 X64 و با استفاده از زبان برنامه‌نویسی Microsoft Visual Basic 6.0 پیاده‌سازی شد.

در قسمت اولِ رهیافت پیشنهادی یعنی الگوریتم ارائه‌شده مبتنی بر بهینه‌سازی کلونی مورچه‌ها در حل مساله ترتیب کارها، در ابتدا تمامی مقادیر فرمونی روی مقدار ناچیز 0.1 تنظیم شدند. ضریب تبخیر 0.998 در نظر گرفته شد که در آزمایشات سریعترین همگرایی را بوجود می‌آورد. پارامترهای تنظیم مقادیر اکتشافی و فوق اکتشافی فرمول (۱۰) یعنی α و β به ترتیب برابر با 1.0 و 0.5 بودند و همچنین تعداد کل مورچه‌ها به 1500 عدد محدود شد.

در قسمت دوم رهیافت پیشنهادی، یعنی حل مساله انتساب کارها به پردازنده‌ها با استفاده از اتوماتای یادگیر سلولی، از الگوریتم $L_{R-\varepsilon P}$ برای آموزش اتوماتا استفاده شد. سریعترین همگرایی‌ها با ضریب تشویق $a = 0.05$ و ضریب تنبیه $b = 0.01 \times a$ بدست آمد. شرط خاتمه تکرار الگوریتم، بدست آمدن 50 زمانبندی یکسان و یا رسیدن به حداکثر تکرار مجاز یعنی 5000 تکرار در نظر گرفته شد.

۶-۲- نتایج قسمت اول رهیافت پیشنهادی

در این بخش، قسمت اول رهیافت پیشنهادی یعنی الگوریتم ارائه‌شده مبتنی بر بهینه‌سازی کلونی مورچه‌ها در حل مساله ترتیب کارها، با سایر رهیافت‌های سنتی موجود مورد مقایسه قرار می‌گیرد. بدیهی است معیار مقایسه در تمامی آزمایشات همانا طول زمانبندی (زمان اتمام کل کارها) بدست‌آمده می‌باشد؛ یعنی در هر آزمایش رهیافتی موفق‌تر معرفی می‌شود که گراف وظایف داده‌شده را بر روی تعداد پردازنده از پیش‌مشخص جوری زمانبندی کند که طول زمانبندی کمتری حاصل شود. می‌توان گفت پس از آنکه رهیافت پیشنهادی ترتیب اجرای کارها را از گراف ورودی استخراج کند؛ این ترتیب با استفاده از روش زودترین زمان اجرای ممکن (که در اکثر زمانبندی‌های لیستی از این همین روش استفاده شده است) به پردازنده‌ها نگاشت می‌شود تا زمان اتمام کل کارها بدست آید که در حقیقت معیار مقایسه عملکرد این رهیافت با سایر رهیافت‌های سنتی موجود است. البته در ادامه، رهیافت جدیدی مبتنی بر اتوماتای یادگیر سلولی جهت انتساب کارها به پردازنده‌ها معرفی می‌شود که می‌تواند انتساب بهتری نسبت به روش EST انجام دهد و بدین ترتیب باعث تقویت عملکرد کلی رهیافت پیشنهادی خواهد شد.

جهت مقایسه و ارزیابی رهیافت پیشنهادی با سایر الگوریتم‌های موجود، از شش گراف وظایف از برنامه‌های واقعی که در جدول (۲) لیست شده‌اند استفاده می‌شود. این گراف‌ها در اکثر تحقیقات مرجع، معیار مقایسه رهیافت‌های موجود بوده‌اند و لذا از اعتبار خاصی برخوردارند.

جدول (۲): گراف‌های وظایف انتخاب‌شده از برنامه‌های واقعی جهت ارزیابی قسمت اول رهیافت پیشنهادی

گراف	مرجع	تعداد گره	هزینه‌های ارتباطی
G1	Kwok and Ahmad [۷]	۹	متغیر
G2	Al-Mouhamed [۳۵]	۱۷	متغیر
G3	Wu and Gajski [۱۹]	۱۸	۴۰ و ۶۰
G4	Al-Maasarani [۳۶]	۱۶	متغیر
G5	گراف وظایف شکل (۲)	۹	متغیر
G6	Hwang *et al.* [۲۷]	۱۸	۸۰ و ۱۲۰

همانطور که در بخش ۵-۱ اشاره شد فرمول (۱۰) که وظیفه اصلی انتخاب حرکات مورچه‌ها را به عهده دارد جهت انتخاب حرکت بعدی توسط هر مورچه از دو معیار استفاده می‌کند. اول مقادیر فوق‌اکتشافی یعنی $\tau_{ij}(t)$ که میزان اثر فرومونی روی یال (n_i, n_j) در تکرار t-ام را نشان می‌دهد که این مقادیر در ابتدا همگی یکسان و نزدیک صفرند و کم‌کم توسط مورچه‌ها بروز می‌شوند. معیار دوم، مقادیر اکتشافی η_j یا همان میزان اولویت کار n_j که می‌توان از معیارهای اولویت مختلفی مانند *TLevel*، *BLevel*، *SLevel*، *ALAP* و *NOO* (معرفی‌شده در بخش۲) به عنوان مقادیر اکتشافی η_j استفاده کرد که بهینه‌ترین آنها باید بصورت آزمایشی مشخص شود.

بدین ترتیب مجموعه آزمایشات اولیه جهت یافتن معیار اولویت مناسب جهت جایگذاری در فرمول (۱۰) انجام شده است. جدول (۳) نتایج حاصل از این آزمایشات را روی هر شش گراف وظایف موجود و با استفاده از دو پردازنده نشان می‌دهد. لازم به ذکر است که از میانگین ۱۰ بار اجرای الگوریتم استفاده شده تا نتایج عادلانه‌تر باشند. با توجه به آنکه گراف‌های استفاده‌شده با هم تفاوت‌های ساختاری فاحشی دارند و طول زمانبندی بدست‌آمده از آنها در بازه وسیعی قرار می‌گیرد؛ لذا طول زمانبندی نرمال‌شده نیز در جدول مذکور ارائه شده است که می‌تواند مرجع مقایسه قرار گیرد. این نرمال‌سازی از طریق فرمول (۱۵) ارائه‌شده در بخش ۶-۴-۲ انجام می‌شود که توضیحات تکمیلی نیز در بخش مذکور

گنجانده شده است. در این مجموعه از آزمایشات استفاده از معیار *ALAP* از نظر میانگین طول زمانبندی نرمال‌شده از سایر معیارهای دیگر موفق‌تر بوده است و در نتیجه در ادامه آزمایشات نیز از این معیار استفاده خواهد شد.

جدول (۳): نتایج حاصل از میانگین ۱۰ بار اجرای الگوریتم روی هر شش گراف وظایف موجود با استفاده از معیارهای اولویت متفاوت

Graph	*CPL*	*TLevel*	*BLevel*	*SLevel*	*ALAP*	*NOO*
G1	11	16.9	16.2	16.1	16.200	16.2
	NSL:	1.536	1.473	1.464	1.473	1.473
G2	28	38	38	38	38	38
	NSL:	1.357	1.357	1.357	1.357	1.357
G3	300	390	390	390	390	390
	NSL:	1.300	1.300	1.300	1.300	1.300
G4	34	47	47	47	47	47
	NSL:	1.382	1.382	1.382	1.382	1.382
G5	12	23.9	23.5	23.4	23.7	23.2
	NSL:	1.992	1.958	1.950	1.975	1.933
G6	300	470	470	459	443	470
	NSL:	1.567	1.567	1.530	1.477	1.567
Average NSL:		**1.522**	**1.506**	**1.497**	**1.494**	**1.502**

جدول (۴): نتایج حاصل از اجرای رهیافت پیشنهادی و سایر رهیافت‌های سنتی روی هر شش گراف وظایف موجود با دو پردازنده

Graph	HLFET	MCP	DLS	ETF	ACO
G1	23	19	21	21	17
G2	44	43	46	44	42
G3	410	420	410	400	390
G4	63	62	60	60	52
G5	30	29	23	23	21
G6	540	550	520	520	440

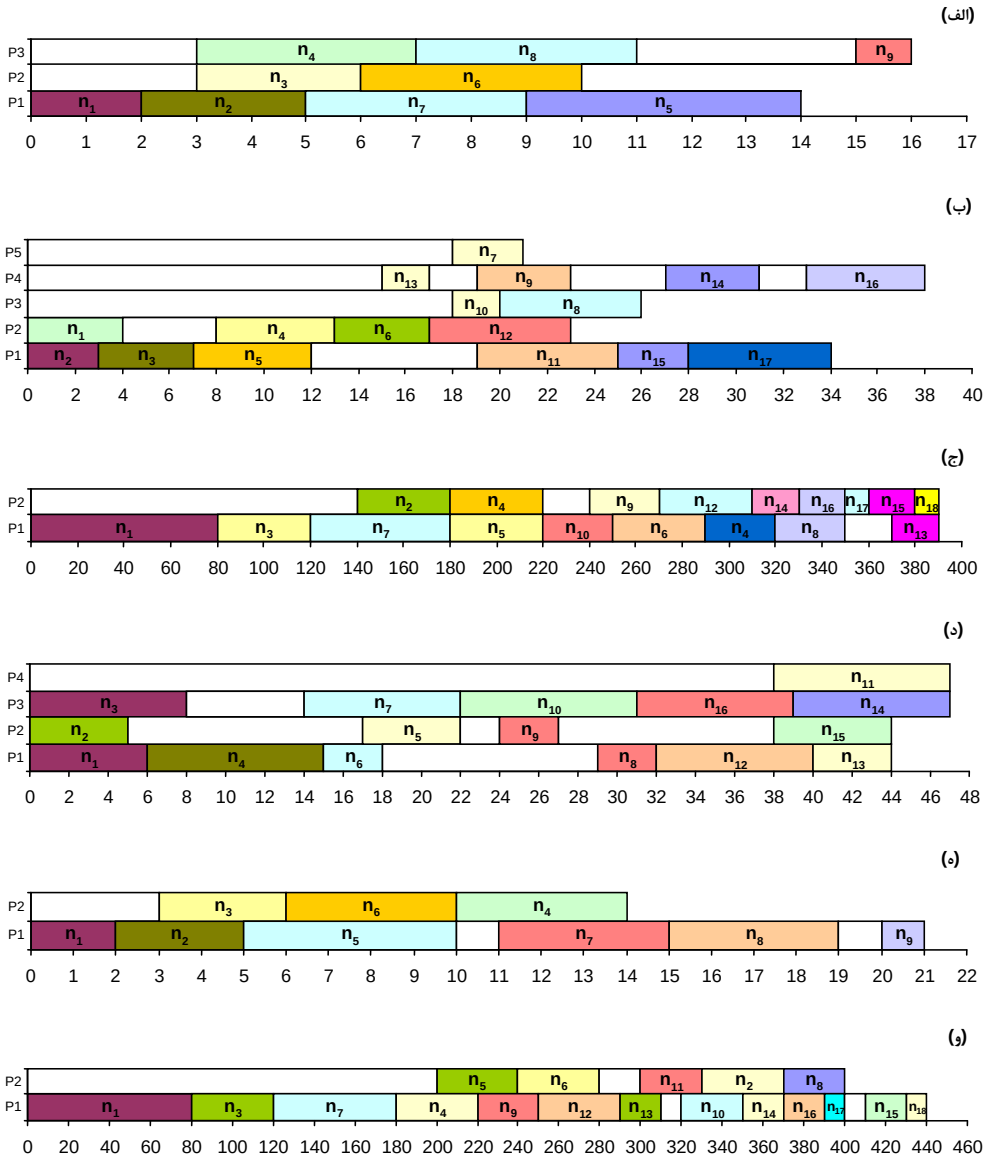
با تعیین معیار اولویت مناسب، حال می‌توان مجموعه آزمایشات بعدی را در جهت ارزیابی عملکرد رهیافت پیشنهادی انجام داد. مجموعه آزمایشات دوم، رهیافت پیشنهادی را با چهار روش سنتی اشاره‌شده یعنی الگوریتم ابتدا بالاترین سطح با تخمین زمان، الگوریتم مسیر بحرانی تعدیل‌شده، الگوریتم زمانبندی سطح پویا و الگوریتم زودترین زمان اجرا اول (که از نظر عملکرد از بهترین روشهای ارائه‌شده در این زمینه هستند) مقایسه می‌کند. در این آزمایشات از هر شش گراف وظایف موجود استفاده شده و البته زمانبندی روی تنها ۲ پردازنده انجام شده است. این آزمایش می تواند یک دید کلی از عملکرد رهیافت پیشنهادی ارائه کند و جهت مقایسه عادلاتر، سایر آزمایشات با تعداد متفاوت پردازنده انجام شده‌اند. جدول (۴) طول زمانبندی حاصل از انجام این آزمایشات را روی گراف‌های وظایف مذکور نشان می‌دهد. همانطور که دیده می‌شود رهیافت پیشنهادی در تمامی موارد موفق به یافتن راه‌حل-های بهتری نسبت به سایر روش‌ها شده است.

استاندارد بودن گراف‌های وظایف انتخاب‌شده، این امکان را مهیا می‌کند تا در مجموعه آزمایشات سوم، رهیافت پیشنهادی با سایر روش‌های ارائه‌شده در این زمینه، چه روش‌های BNP و چه روش-های UNC مقایسه گردد. همانگونه که اشاره شد در روش‌های UNC تعداد پردازنده از قبل مشخص نیست و هر تعداد دلخواه می‌تواند باشد. لذا در این آزمایش تعداد پردازنده‌ها نامحدود در نظر گرفته شده تا هر الگوریتم دقیقاً بهترین راه‌حل ممکنه را بیابد. نتایج این آزمایشات در جدول (۵) نشان داده شده است. اینجا نیز رهیافت پیشنهادی در تمامی موارد بهترین راه‌حل‌ها را یافته است و فقط در معدود موارد برخی از سایر الگوریتم‌ها عمکردی مساوی با رهیافت پیشنهادی داشته‌اند و این خود نشاندهنده برتری رهیافت پیشنهادی از نظر عملکرد نسبت به سایر روش‌های موجود است. همچنین نمودار گانت بهترین زمانبندی یافته‌شده توسط رهیافت پیشنهادی روی هر شش گراف وظایف در شکل (۶) به نمایش گذاشته شده است که می‌تواند جهت حصول اطمینان از صحت اجرای این رهیافت مورد استفاده قرار گیرد.

در ادامه مجدداً استاندارد بودن گراف‌های وظایف انتخاب‌شده این امکان را می‌دهد تا رهیافت پیشنهادی با یکی از بهترین روش‌های مبتنی بر الگوریتم ژنتیکِ ارائه‌شده در این زمینه مقایسه شود. جدول (۶) نتایج حاصل از این آزمایشات را روی گراف‌های G5 و

جدول (۵): نتایج حاصل از اجرای رهیافت پیشنهادی و سایر رهیافت‌های سنتی موجود اعم از BNP و UNC روی هر شش گراف وظایف موجود با تعداد پردازنده نامحدود

Graph	LC	EZ	MD	DSC	DCP	HLFET	ISH	ETF	LAST	MCP	DLS	ACO
G1	19	18	17	-	16	19	19	19	19	20	19	16
G2	39	40	38	38	38	41	38	41	43	40	41	38
G3	420	540	420	390	390	390	390	390	470	390	390	390
G4	-	-	-	-	-	48	-	48	-	48	47	47
G5	-	-	32	27	23	29	-	29	-	29	29	21
G6	-	-	460	460	440	520	-	520	-	520	520	440



شکل (۶): نمودار گانت بهترین زمانبندی یافته‌شده توسط رهیافت پیشنهادی روی هر شش گراف وظایف موجود الف) گراف G1 با ۹کار ب) گراف G2 با ۱۷کار ج) گراف G3 با ۱۸کار د) گراف G4 با ۱۶کار ه) گراف G5 با ۹کار و) گراف G6 با ۱۸کار

G6 نشان می‌دهد. همانطور که مشاهده می‌شود رهیافت پیشنهادی برای گراف اولی بهترین راه‌حل را بدست آورده و عمکردش برای گراف دومی معادل الگوریتم ژنتیک است و این در حالیست که در الگوریتم ژنتیک مذکور هر نسل دارای ۱۰۰ کرومزوم بوده و تعداد نسل‌ها نیز برابر ۱۰۰۰ می‌باشد یعنی این الگوریتم ژنتیک با آزمایش ۱۰۰،۰۰۰ راه حل، جواب نهایی را می‌یابد در حالی که رهیافت پیشنهادی تنها از ۱۵۰۰ مورچه استفاده می‌کند بدین معنا که تنها با آزمایش ۱۵۰۰ راه‌حل جواب نهایی را ارائه می‌کند! به عبارت دیگر رهیافت پیشنهادی جواب نهایی را بسیار سریعتر از الگوریتم ژنتیک می‌یابد. البته این امر منطقی است زیرا در الگوریتم بهینه‌سازی کلونی مورچه‌ها، ارتباط غیرمستقیم محلی به نام استیگمرجی بین مورچه وجود دارد و هر مورچه با استفاده از تجربیات تمامی مورچه‌های قبلی مسیر خود را انتخاب می‌کند در حالی که در الگوریتم ژنتیک بسیاری از تجربیات بدست‌آمده از طریق حذف کرمزوم‌های متوسط و ضعیف از بین می‌رود.

جدول (۶): نتایج بدست‌آمده از رهیافت پیشنهادی روی گراف‌های G5 و G6 در کنار رهیافت مبتنی بر الگوریتم ژنتیک و چند رهیافت سنتی

Graph	MCP	DSC	MD	DCP	Genetic	ACO
G5	29	27	32	32	23	21
G6	520	460	460	440	440	440

پیاده‌سازی‌ها با استفاده از ماتریس مجاورتی گراف نشان می‌دهند که پیچیدگی زمانی قسمت اول رهیافت پیشنهادی در حل زیرمساله ترتیب کارها برابر ($O(No\_of\_Ants \times n^2)$ می‌باشد که *No_of_Ants* برابر تعداد کل مورچه‌های ایجادشده بوده و در رهیافت پیشنهادی برابر ۱۵۰۰ در نظر گرفته شده است؛ در نتیجه در گراف‌های وظایف به اندازه کافی بزرگ که بتوان از این ثابت عددی صرفه‌نظر کرد رهیافت پیشنهادی اجرای برابر یا سریعتری نسبت به روش‌های سنتی چون HLFET، MCP، DLS و ETF خواهد داشت که به ترتیب دارای پیچیدگی زمانی ($O(n^2)$، ($O(n^3)$، ($O(mn^2)$ و ($O(mn^2)$ می‌باشند. درحقیقت می‌توان گفت تنها نقطه ضعف این قسمت از رهیافت پیشنهادی در سربار محاسباتی اضافی است که اجرای الگوریتم بهینه‌سازی کلونی مورچه‌ها در زمانبندی گراف‌های وظایف کوچک بر سیستم اعمال می‌کند. این مشکل با ایجاد مصالحه بین اندازه گراف ورودی و تعداد مورچه‌های ایجادشده بر اساس آن حل نخواهد شد؛ زیرا ماهیت این الگوریتم در کار گروهی و همکاری بین تعداد زیاد مورچه نهفته بوده و در استفاده از تعداد قلیل از مورچه‌ها، همگرایی لازم ایجاد نخواهد شد.

۶-۳- نتایج قسمت دوم رهیافت پیشنهادی

در این بخش، قسمت دوم رهیافت پیشنهادی یعنی الگوریتم ارائه شده مبتنی بر اتوماتای یادگیر سلولی در حل مساله انتساب کارها، با روش سنتی موجود یعنی روش زودترین زمان اجرای ممکن (که در اکثر زمانبندی‌های لیستی از این روش استفاده شده است) مقایسه می‌شود. در این راستا، ابتدا چهار گراف وظایف از برنامه‌های واقعیِ مختلف در نظر گرفته شده که در مراجع معتبر جهت مقایسه اینگونه الگوریتم‌ها بکار رفته‌اند. این گراف‌ها و توضیحاتشان را در جدول (۷) مشاهده می‌کنید.

با توجه به آنکه زمانبندی بدست‌آمده از ترتیب‌های مختلف کارها می‌تواند متفاوت باشد و این تفاوتِ ترتیب، می‌تواند رهیافت پیشنهادی را به چالش بکشد. لذا برای هر گراف با توجه به اولویت‌های متفاوت *TLevel*، *BLevel*، *SLevel*، *ALAP* و *NOO* و به روش زمانبندی لیستی، پنج ترتیب اجرای مختلف استخراج می‌شود. ترتیب بدست‌آمده از معیارهای *TLevel* و *ALAP* گره‌ها را به ترتیب صعودیِ معیار و ترتیب بدست‌آمده از معیارهای *BLevel*، *SLevel* و *NOO* گره‌ها را به ترتیب نزولیِ معیار مرتب می‌کنند. جدول (۸) ترتیب‌های استخراج‌شده از گراف وظایف شکل (۲) را با توجه به در نظرگرفتن معیارهای اولویتِ مختلف نشان می‌دهد. جهت بازرسی صحت این ترتیب‌ها می‌توان از اعداد موجود در جدول (۱) نیز استفاده کرد.

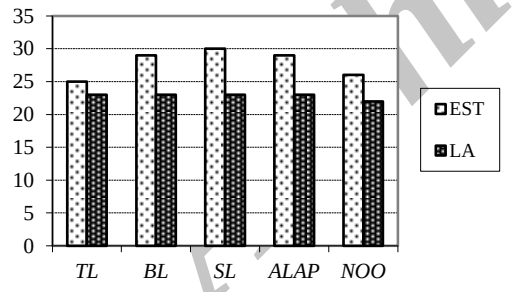
جدول (۷): گراف‌های وظایف انتخاب‌شده از برنامه‌های واقعی جهت ارزیابی قسمت دوم رهیافت پیشنهادی

گراف	مرجع	تعداد گره	هزینه‌های ارتباطی
G1	Al-Mouhamed [۳۵]	۱۷	متغیر
G2	Wu and Gajski [۱۹]	۱۸	۴۰ و ۶۰
G3	گراف وظایف شکل (۲)	۹	متغیر
G4	Hwang *et al.* [27]	۱۸	۸۰ و ۱۲۰

جدول (۸): ترتیب گره‌های گراف وظایف شکل (۲) با توجه به در نظرگرفتن معیارهای اولویت مختلف

TLevel	n_1	n_3	n_4	n_5	n_2	n_6	n_8	n_7	n_9
BLevel	n_1	n_5	n_2	n_3	n_4	n_6	n_7	n_8	n_9
SLevel	n_1	n_5	n_4	n_2	n_3	n_6	n_7	n_8	n_9
ALAP	n_1	n_5	n_2	n_3	n_4	n_6	n_7	n_8	n_9
NOO	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9

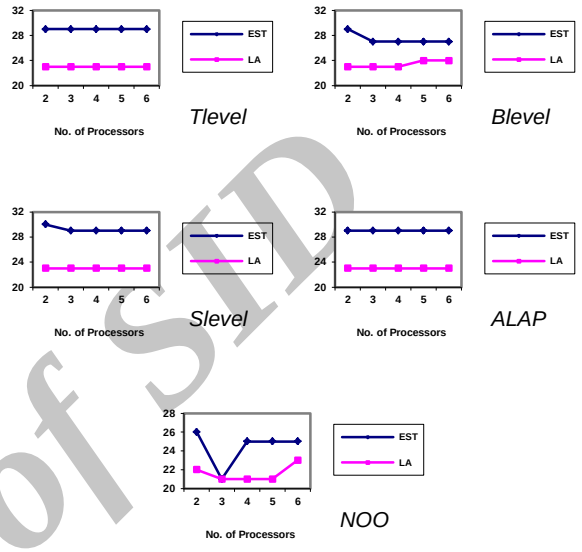
نمودار شکل (۷) زمان اتمام بدست‌آمده از روش زودترین زمان ممکن (EST) و روش پیشنهادی اول مبتنی بر اتوماتای یادگیر (LA) را با توجه به ترتیب‌های اجرای مختلف از معیارهای ذکرشده و مندرج در جدول (۸) روی تنها دو پردازنده نشان می‌دهد. همانگونه که مشاهده می‌شود رهیافت پیشنهادی در هر پنج ترتیب متفاوت که از معیارهای مختلف بدست آمده‌اند نسبت به روش سنتیِ زودترین زمان اجرای ممکن، انتساب بهتری انجام داده و زمان اتمام کمتری بدست آورده است. ناگفته پیداست که نتایج این آزمایش نشان می‌دهد که روش سنتی EST بهینه نیست.



شکل (۷): مقایسه زمان اتمام بدست‌آمده از روش زودترین زمان ممکن و روش پیشنهادی اول مبتنی بر اتوماتای یادگیر برای ترتیب‌های مختلف گراف وظایف شکل (۲) روی دو پردازنده

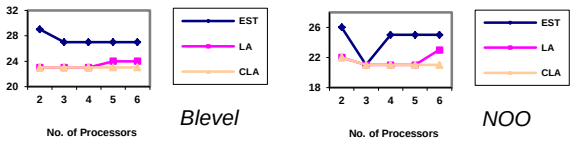
مجموعه آزمایشات بعدی رفتار روش پیشنهادی را در تقابل با افزایش تعداد پردازنده‌ها از دو پردازنده تا شش پردازنده ارزیابی می‌کند. نتایج این آزمایشات روی ترتیب‌های مختلف ذکرشده در جدول (۸) به روش پیشنهادی و روش سنتیِ EST در شکل (۸) نشان داده شده است. هر چند انتظار می‌رود با افزایش تعداد پردازنده‌ها، زمانِ اتمام برنامه کاهش یابد اما لزوماً اینچنین نیست.

به عنوان یک قانون کلی هر چه کارها روی پردازنده‌های کمتری زمانبندی شوند باید هزینه‌های ارتباطی کمتری بپردازند. در مقابل با وجود پردازنده‌های بیشتر، زمان شروع سریعتری می‌توانند داشته باشند. بنابراین می‌توان گفت اگر نسبتِ زمان لازم جهت اجرای کارها به هزینه‌های ارتباطی، بالا باشد پردازنده‌ی بیشتر موثر است و اگر این نسبت پایین باشد زمانبندی فشرده‌تر جواب بهتری می‌دهد.



شکل (۸): زمانبندی بدست‌آمده از روش پیشنهادی اول و روش سنتیِ زودترین زمان ممکن با توجه به ترتیب‌های مختلف از معیارهای متفاوت در جدول (۸) روی دو تا شش پردازنده

مشکل اصلی رهیافت پیشنهادی مبتنی بر اتوماتای یادگیر در افزایش تعداد پردازنده‌هاست. همانگونه در شکل (۸) برای زمانبندی ترتیب‌های *BLevel* و *NOO* دیده می‌شود با افزایش تعداد پردازنده‌ها، زمانبندی بدست‌آمده از رهیافت پیشنهادی بدتر می‌شود به این دلیل که با افزایش تعداد پردازنده‌ها، فضای حالات مساله بسیار بزرگتر شده و هر اتوماتا عمل‌های مجاز بیشتری جهت انتخاب در اختیار دارد. لذا در اکثر موارد همگرایی لازم صورت نگرفته و الگوریتم پس از حداکثر تکرار خود یعنی ۵۰۰۰ تکرار متوقف می‌شود. جهت رفع این مشکل از رهیافت دومی مبتنی بر اتوماتای یادگیر سلولی نامنظم استفاده شده است. در این رهیافت یک قانون محلی وجود دارد مبنی بر اینکه اگر یک سلول پردازنده‌ای را جهت اجرا انتخاب کند که تمامی همسایگان آن سلول (تمام پدران آن کار) انتخاب کرده‌اند آنگاه آن سلول پاداش می‌گیرد و در غیر اینصورت تنبیه می‌گردد. با وجود این قانون، اتوماتا همواره سعی می‌کند در زمانبندی از تعداد پردازنده کمتری استفاده کرده و زمانبندی فشرده‌تری انجام دهد که در بسیاری موارد منجر به تولید زمانبندی بهتر و همگرایی زودتری خواهد شد. شکل (۹) زمانبندی بدست‌آمده از ترتیب‌های *BLevel* و *NOO* را برای هر دو رهیافت پیشنهادی و با توجه به افزایش تعداد پردازنده‌ها نشان می‌دهد. نتایج موجود در این شکل نشان می‌دهد که رهیافت دوم مبتنی بر اتوماتای یادگیر سلولی نامنظم در پردازنده‌های بالا، جواب‌های بهتری نسبت به رهیافت اول می‌یابد.

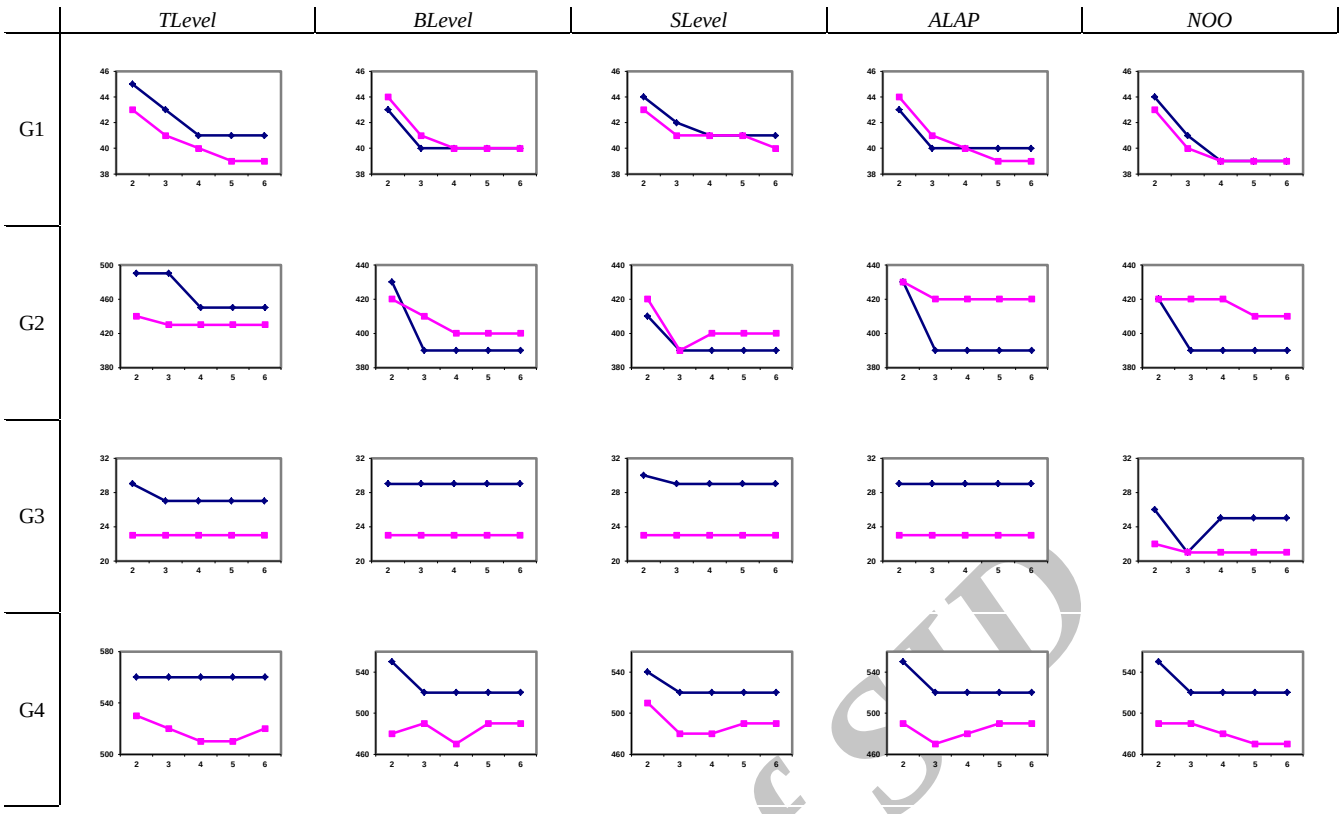


شکل (۹): زمانبندی بدست‌آمده از ترتیب‌های *SLevel* و *NOO* برای هر دو رهیافت پیشنهادی با توجه به افزایش تعداد پردازنده‌ها

بدین ترتیب مجموعه آزمایشات بعدی با استفاده از رهیافت دوم مبتنی بر اتوماتای یادگیر سلولی انجام شد. شکل (۱۰) نتایج زمانبندی بدست‌آمده از گراف‌های وظایف موجود در جدول (۷) را با استفاده از روشِ سنتی زودترین زمان شروع ممکن و روش پیشنهادی دوم مبتنی بر اتوماتای یادگیر سلولی نشان می‌دهد. نتایج نشاندهنده آنست که در اکثر موارد رهیافت پیشنهادی زمانبندی بهتری نسبت به روش سنتی بدست آورده و می‌تواند انتساب بهینه‌تری انجام دهد.

۶-۴- نتایج کلی حاصل از رهیافت پیشنهادی (ادغام قسمت اول و دوم)

در بخش ۶-۲ نتایج بدست‌آمده از قسمت اول رهیافت پیشنهادی یعنی حل مساله ترتیب کارها با استفاده از الگوریتم بهینه‌سازی کلونی مورچه‌ها مورد ارزیابی قرار گرفت. دیدیم که این قسمت از رهیافت پیشنهادی از نظر عملکرد از بسیاری از روش‌های سنتی و تکاملی موجود مانند الگوریتم ابتدا بالاترین سطح با تخمین زمان، الگوریتم مسیر بحرانی تعدیل‌شده، الگوریتم زمانبندی سطح پویا، الگوریتم زودترین زمان اجرا اول و الگوریتم ژنتیک بهتر بوده و زمانبندی‌های بهتری می‌یابد. همچنین در بخش ۶-۳ عملکرد بخش دوم رهیافت پیشنهادی یعنی حل مساله انتساب با استفاده از اتوماتای یادگیر سلولی مورد بحث و بررسی قرار گرفت و همچنان که مشاهده شد این قسمت نیز از روش سنتی انتساب کارها به پردازنده ها که EST نام داشت بهتر عمل کرده و انتساب-های مطلوبتری انجام می‌داد. در این بخش با در کنار هم قرار



شکل (۱۰): نتایج زمانبندی بدست‌آمده از گراف‌های وظایف موجود در جدول (۷) با استفاده از روشِ سنتی زودترین زمان شروع ممکن و روش پیشنهادی مبتنی بر اتوماتای یادگیر سلولی

گرفتن قسمت اول و دوم رهیافت پیشنهادی عملکرد کلی سیستم سنجیده می‌شود. بدین ترتیب که ابتدا الگوریتم بهینه‌سازی کلونی مورچه، ترتیب صحیح کارها را از گراف وظایف ورودی استخراج کرده و سپس این ترتیب با استفاده از اتوماتای یادگیر سلولی به پردازنده‌های موجود نگاشت می‌شود تا زمانبندی نهایی بدست آید.

۶-۴-۱- پایگاه داده مورد استفاده

جهت ارزیابی قسمت‌های اول و دوم رهیافت پیشنهادی از گراف-های وظایف استاندارد مستخرج از برنامه‌های واقعی استفاده شد که در بسیاری از تحقیقات، مرجع استفاده بودند. در هر دو قسمت، رهیافت پیشنهادی نسبت به روش‌های سنتی موجود عملکرد بهتری داشت که به تبع برآیند حاصل از ترکیب قسمت‌های اول و دوم رهیافت پیشنهادی نیز از روش‌های موجود برتری خواهد جست. لذا جهت ارزیابی بیشتر و عادلانه‌تر، در ادامه از مجموعه‌ای از ۱۲۵ گراف وظایف تصادفی استفاده خواهد شد که از لحاظ معیارهای ساختاری زیر متفاوتند:

- اندازه گراف وظایف[1]: که در حقیقت همان تعداد گره (کار) در گراف وظایف می‌باشد. پنج اندازه مختلف از این معیار استفاده شده است {۳۲، ۶۴، ۱۲۸، ۲۵۶ و ۵۱۲}.
- ضریب ارتباط به محاسبه[2]: که نشان می‌دهد هر گراف تا چه حد مبتنی بر محاسبه یا هزینه های ارتباطی است. اگر در گرافی هزینه‌های ارتباطی بین کارها بالا بوده و زمان لازم جهت محاسبه کارها کم باشد آنگاه این ضریب بالا است و در غیر اینصورت این ضریب برای گراف مزبور پایین است. وزن هر گره با توزیع یکنواختی بصورت تصادفی با میانگین ۵۰ انتخاب شده و هزینه‌های ارتباطی روی یال‌ها نیز با توزیع یکنواخت و بصورت تصادفی با میانگین وزن گره‌ها × CCR انتخاب می‌شود. پنج مقدار مختلف {۰,۱، ۰,۵، ۱,۰، ۵,۰ و ۱۰,۰} برای این معیار در نظر گرفته شده که انتخاب ۰,۱ گراف حاصل را مبتنی بر محاسبه و انتخاب ۱۰,۰ گراف حاصل را مبتنی بر ارتباط خواهد کرد.

[1] Size(n)
[2] Communication-to-Computation-Ratio (CCR)

- معیار توازی[1]: که نشاندهنده میانگین تعداد فرزند برای هر گره است. افزایش مقدار این معیار، درخت را متصل‌تر و کاهش این معیار درخت را هرس می‌کند. پنج مقدار مختلف {۳، ۵، ۱۰، ۱۵ و ۲۰} برای این معیار انتخاب شده است.

۶-۴-۲- ارزیابی نتایج رهیافت پیشنهادی

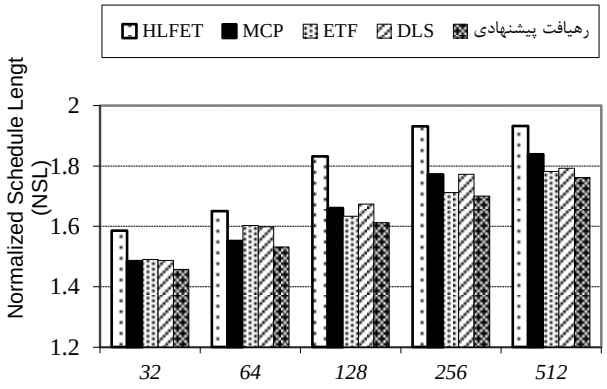
از آنجا که گراف‌های وظایف تصادفی که جهت ارزیابی رهیافت پیشنهادی انتخاب شده‌اند طیف وسیعی دارند و نتایج بدست‌آمده از زمانبندی آنها با استفاده از رهیافت پیشنهادی و سایر الگوریتم-های موجود در بازه وسیعی قرار می‌گیرد؛ لذا جهت ارزیابی عملکرد، از معیار استاندارد جدیدی به نام طول زمانبندی نرمال-شده[2] استفاده می‌شود که معادلِ تقسیمِ زمان نهایی بدست‌آمده بر وزن مسیر بحرانی موجود در گراف وظایف ورودی بوده و از طریق فرمول زیر بدست می‌آید.

$$NSL = \frac{Schedule\ \ Length}{\sum_{n_i \in CP} w_i} \qquad (۱۵)$$

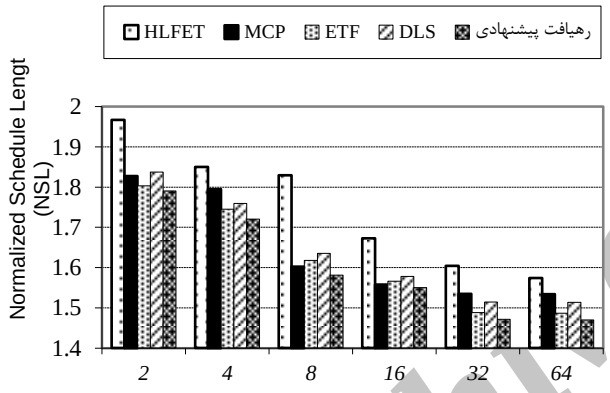
که Schedule Length همان زمان نهایی بدست‌آمده از گراف وظایف ورودی و CP همان مجموعه گره‌های موجود در مسیر بحرانی است (مسیر بحرانی هر گراف طولانی‌ترین مسیر از گره ورودی به گره خروجی بدون احتساب هزینه‌های ارتباطی است [۷]). همچنین با توجه به آنکه رهیافت پیشنهادی یک الگوریتم BNP است که تعداد پردازنده‌ها از پیش مشخص بوده و تکثیر کارها نیز مجاز نمی‌باشد لذا جهت ارزیابی عادلانه، از همان چهار رهیافت سنتی معرفی‌شده در بخش۲ یعنی الگوریتم ابتدا بالاترین سطح با تخمین زمان، الگوریتم مسیر بحرانی تعدیل‌شده، الگوریتم زمانبندی سطح پویا، الگوریتم زودترین زمان اجرا اول استفاده شده است.

شکل (۱۱) میانگین طول زمانبندی نرمال‌شده را برای تمامی ۱۲۵ گراف تصادفی ورودی با توجه به اندازه مختلف گراف‌ها نشان می-دهد. بدیهی است با افزایش اندازه گراف، معیار مورد نظر نیز افزایش یافته است. همانطور که مشاهده می‌شود رهیافت پیشنهادی در تمامی اندازه‌ها، عملکرد بهتری نسبت به سایر روش-ها دارد.

شکل (۱۲) نیز میانگین طول زمانبندی نرمال‌شده را برای تمامی ۱۲۵ گراف تصادفی ورودی با توجه به تعداد مختلف پردازنده جهت انتساب نشان می‌دهد. بدیهی است با افزایش تعداد پردازنده، می‌توان زمانبندی‌های بهتری انجام داد و بدین ترتیب معیار مورد نظر کاهش یافته است. همانطور که مشاهده می‌شود مجدداً رهیافت پیشنهادی در تمامی موارد، عملکرد بهتری نسبت به سایر روش‌ها داشته و از آنها پیشی گرفته است.



شکل (۱۱): میانگین طول زمانبندی نرمال‌شده برای تمامی ۱۲۵ گراف تصادفی ورودی با توجه به اندازه مختلف گرافها



شکل (۱۲): میانگین طول زمانبندی نرمال‌شده برای تمامی ۱۲۵ گراف تصادفی ورودی با توجه به تعداد مختلف پردازنده جهت انتساب

۷- نتیجه گیری

زمانبندی کارها یکی از مهمترین چالش‌ها در رسیدن به عملکرد بالا در سیستم‌های چند پردازنده‌ای چون سیستم‌های موازی و توزیع شده است. از آنجائیکه این مساله جزء مسائل با پیچیدگی زمانی سخت می‌باشد؛ لذا استفاده از روش‌های اکتشافی و فوق-اکتشافی در حل این مساله بسیار منطقی است. روش‌های ارائه‌شده جهت حل این مساله به سه دسته اصلی BNP، UNC و TDB تقسیم می‌شوند که دسته اول با توجه به آنکه تکرار کارها را مجاز ندانسته و تعداد پردازنده‌ها از پیش مشخص است در کاربردهای واقعی بسیار مرسوم‌تر است.

الگوریتم‌های سنتیِ ارائه‌شده در این دسته، جهت زمانبندی از روشی به نام زمانبندی لیستی استفاده می‌کنند. بدین ترتیب که در ابتدا با استفاده از معیارهایی، لیستی از کارها به ترتیب اولویت

[1] Parallelism
[2] Normalized Schedule Length (*NSL*)

ساخته شده و سپس مرحله به مرحله، پراولویت‌ترین کار درون لیست خارج شده و به پردازنده‌ای که زودتر می‌تواند آن را اجرا کند اختصاص می‌یابد. تا در نهایت کل کارها از لیست خارج شوند و زمانبندی نهایی ایجاد شود. در این روش همواره دو عامل اصلی، طول زمانبندی بدست‌آمده از رهیافت‌های مختلف را تحت شعاع قرار می‌دهد. اول اینکه کارها به چه ترتیبی جهت اجرا انتخاب شوند (زیرمساله ترتیب) و دوم این مساله که ترتیب انتخاب‌شده جهت اجرا، چگونه به پردازنده‌ها نگاشت شود (زیرمساله انتساب).

در رهیافت پیشنهادیِ این تحقیق که خود از دسته اول یعنی BNP می‌باشد. الگوریتم بهینه‌سازی کلونی مورچه‌ها ترتیب اجرای کارها را مشخص کرده و مساله ترتیب کارها را حل می‌کرد و اتوماتای یادگیر سلولی با حل مساله انتساب، ترتیبِ کارهایِ مشخص‌شده را روی پردازنده‌ها نگاشت می‌کرد.

جهت ارزیابی رهیافت پیشنهادی، ابتدا قسمت اول رهیافت، یعنی حل مساله ترتیب کارها با استفاده از الگوریتم بهینه‌سازی کلونی مورچه ها مورد توجه قرار گرفت. جهت مقایسه از چهار روش سنتی از بهترین الگوریتم‌های BNP با نام‌های الگوریتم ابتدا بالاترین سطح با تخمین زمان، الگوریتم مسیر بحرانی تعدیل‌شده، الگوریتم زمانبندی سطح پویا، الگوریتم زودترین زمان اجرا اول استفاده شد. گراف‌های ورودی متشکل از شش گراف وظایف استاندارد از برنامه‌های واقعی مختلف بود که در بسیاری از تحقیقات انجام گرفته مرجع ارزیابی بودند. آزمایشات مختلفی با توجه به تعداد پردازنده‌های متفاوت انجام شد که در تمامی موارد رهیافت پیشنهادی موفق به یافتن بهترین زمانبندی‌ها شد و از نظر عملکرد از رقبای سنتی خود پیشی گرفت. همچنین این قسمت از رهیافت پیشنهادی با یکی از بهترین الگوریتم‌های ژنتیک ارائه‌شده در این زمینه نیز مقایسه شد. این مقایسه بر روی دو گراف وظایف مختلف انجام شد که رهیافت پیشنهادی یکبار زمانبندی بهتر یافت و زمانبندی یافته شده برای گراف دوم معادل جوابِ الگوریتم ژنتیک بود و این در حالیست که رهیافت پیشنهادی بسیار سریعتر عمل می‌کند. بطوریکه رهیافت پیشنهادی فقط ۱۵۰۰ راه حل را مقایسه می‌کرد ولی الگوریتم ژنتیک از ۱۰۰،۰۰۰ راه حل سود می‌بُرد.

آزمایشات بعدی جهت مقایسه قسمت دوم الگوریتم، یعنی حل مساله انتساب کارها به پردازنده‌ها با استفاده از اتوماتای یادگیر و اتوماتای یادگیر سلولی در مقابل تنها روش سنتی موجود یعنی روش زودترین زمان اجرای ممکن انجام شد. در این بخش ابتدا رهیافتی مبتنی بر اتوماتای یادگیر ارائه شد که در بعضی موارد بجای بهبود زمانبندی نهایی، آنرا طولانی‌تر کرده و راه‌حل‌های نامناسبی ارائه می‌کرد که این نقیصه با استفاده از اتوماتای یادگیر سلولی و چند قانون محلی مرتفع گردید. آزمایشات بسیاری روی چهار گراف وظایف از برنامه‌های واقعی مرجع انجام شد که بطور میانگین رهیافت پیشنهادی عملکرد بهتری داشت و این خود ثابت میکرد که روش EST یک روش بهینه نیست. این مساله از جمله دستآوردهای مهم این تحقیق می‌باشد.

در نهایت با در کنار هم قراردادن قسمت اول و دوم رهیافت پیشنهادی عملکرد کلی سیستم مورد سنجش قرار گرفت. بدین ترتیب که ابتدا الگوریتم بهینه‌سازی کلونی مورچه‌ها، ترتیب صحیح کارها را از گراف وظایف ورودی استخراج کرده و سپس این ترتیب با استفاده از اتوماتای یادگیر سلولی به پردازنده‌های موجود نگاشت می‌شد و زمانبندی نهایی بدست می‌آمد. در آزمایشات نهایی، جهت ارزیابی بیشتر و عادلانه‌ترِ رهیافت پیشنهادی، از ۱۲۵ گراف وظایف تصادفی استفاده شد که از لحاظ معیارهای ساختاری چون اندازه گراف وظایف (تعداد گره های گراف)، ضریب ارتباط به محاسبه (نسبت متوسط وزن یال‌ها به وزن گره‌ها) و معیار توازی (میانگین تعداد فرزندان هر گره) متفاوت بودند. از آنجا که گراف-های وظایف تصادفی که جهت ارزیابی رهیافت پیشنهادی انتخاب شده‌اند طیف وسیعی داشتند و نتایج بدست‌آمده از زمانبندی آنها با استفاده از رهیافت پیشنهادی و سایر الگوریتم‌های موجود در بازه وسیعی قرار می‌گرفت؛ لذا جهت مقایسه از معیار استاندارد جدیدی به نام طول زمانبندی نرمال‌شده استفاده شد که معادلِ تقسیم زمان نهایی بدست‌آمده بر وزن مسیر بحرانی موجود در گراف وظایف ورودی بود. مجدداً جهت مقایسه از همان الگوریتم-های ابتدا بالاترین سطح با تخمین زمان، الگوریتم مسیر بحرانی تعدیل‌شده، الگوریتم زمانبندی سطح پویا و الگوریتم زودترین زمان اجرا اول استفاده شد و در نهایت رهیافت پیشنهادی در تمامی موارد عملکرد بهتری ارائه نمود.

سپاسگزاری

بر خود لازم میدانم از آموزشکده فنی و حرفه ای سما واحد شوشتر که حمایت مالی این تحقیق را به عهده داشت و تمامی عزیزانی که از رهنمون‌های دقیقشان استفاده بردم و همچنین استاد جلیل‌القدر جناب آقای دکتر امیرمسعود رحمانی که این حوزه پژوهشی را نزد ایشان طلبگی نمودم؛ صمیمانه تشکر و قدردانی کنم.

منابع

[۱]. س. پارسا، ش. لطفی و ن. لطفی، «رویکردی مبتنی بر پردازش تکاملی برای زمانبندی گراف وظایف در معماری چندپردازنده‌ای»، *یازدهمین کنفرانس بین‌المللی کامپیوتر ایران*، صفحات ۶۲۷-۶۳۴، تهران، ۱۳۸۴.

[۲]. م. سلمانی، م. زالی و م. مقیمی، «زمانبندی وظایف سیستم‌های چندپردازنده‌ای با کمک یادگیری تقویتی و الگوریتم ژنتیک»، *دوازدهمین کنفرانس بین‌المللی کامپیوتر ایران*، صفحات ۱۹۴۸-۱۹۵۱، تهران، ۱۳۸۵.

[۳]. م. عبدیزدان و ا. رحمانی، «زمانبندی کارها در سیستم‌های چندپردازنده‌ای با استفاده از یک الگوریتم جدید اولویت بر اساس تعداد فرزندان»، *سیزدهمین کنفرانس بین‌المللی کامپیوتر ایران*، کیش، ۱۳۸۶.

[۴]. مصطفی ماهی و پریسا امین نژاد، «الگوریتم ژنتیک ترکیبی زمانبندی گراف وظایف در معماری چند پردازنده ای»، *چهارمین کنفرانس مهندسی برق و الکترونیک ایران*، گناباد، دانشگاه آزاد اسلامی واحد گناباد، ۱۳۹۱.

[۵]. هادی لطفی و بابک آقامحمدی، «زمانبندی گراف وظایف در سامانه های چند پردازنده ای ناهمگن با استفاده از الگوریتم ژنتیک دانه درشت»، *همایش مشترک مهندسی کامپیوتر و مکانیک*، میاندوآب، دانشگاه جامع علمی و کاربردی مرکز میاندوآب، ۱۳۹۲.

[6]. P. Chretienne and *et al., Scheduling Theory and Its Application*, Wiley, New York, 1995.

[7]. Y. Kwok and I. Ahmad, *Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors*, Final report, Hong Kong Research Grants Council, Report No.: HKUST 734/96E and HKUST 6076/97E, Hong Kong, 1998.

[8]. I. Ahmad and Y. Kwok, "On Parallelizing the Multiprocessor Scheduling Problem," IEEE Trans. Parallel Distrib. Syst., vol. 10, no. 8, pp. 795-18, 1999.

[9]. CH. Papadimitriou and M. Yannakakis, "Scheduling Interval-Ordered Tasks," SIAM J. Computing, vol. 8, pp. 405-5, 1979.

[10]. B. Kruatrachue and TG. Lewis, *Duplication Scheduling Heuristics (DSH): A New Precedence Task Scheduler for Parallel Processor Systems*, Final report, Oregon State University, Report No.: OR 97331, Corvallis, 1987.

[11]. JY. Colin and P. Chretienne, "C.P.M. Scheduling with Small Computation Delays and Task Duplication," Operations Research, vol. 39, no. 4, pp. 680-5, 1991.

[12]. YC. Chung and S. Ranka "Application and Performance Analysis of a Compile-Time Optimization Approach for List Scheduling Algorithms on Distributed-Memory Multiprocessors," In: Proceeding of the IEEE/ACM conference on Supercomputing, pp. 512-10, Nov. 1992.

[13]. H. Chen, B. Shirazi and J. Marquis, "Performance Evaluation of A Novel Scheduling Method: Linear Clustering with Task Duplication," In: Proceeding of the Int'l Conf. Parallel and Distributed Systems, pp. 270-6, Dec. 1993.

[14]. I. Ahmad and YK. Kwok, "On Exploiting Task Duplication in Parallel Program Scheduling," IEEE Trans. Parallel and Distributed Systems, vol. 9, no. 9, pp. 872-21, 1998.

[15]. MA. Palis, JC. Liou and DSL. Wei, "Task Clustering and Scheduling for Distributed Memory Parallel Architectures," IEEE Trans. Parallel and Distributed Systems, vol. 7, no. 1, pp. 46-10, 1996.

[16]. GL. Park, B. Shirazi and J. Marquis, "DFRN: A New Approach for Duplication Based Scheduling for Distributed Memory Multiprocessor Systems," In: Proceeding of the 11th Int'l IEEE Symposium on Parallel Processing, pp. 157-10, Apr. 1997.

[17]. SJ. Kim and JC. Browne, "A General Approach to Mapping of Parallel Computation upon Multiprocessor Architectures," In: Proceeding of the 1988 Int'l Conference on Parallel Processing, vol. 2, pp. 1-8, Aug. 1988.

[18]. V. Sarkar, *Partitioning and Scheduling Parallel Programs for Multiprocessors*, MIT Press, Cambridge (MA), 1989.

[19]. MY. Wu and DD. Gajski, "Hypertool: A Programming Aid for Message-Passing Systems," IEEE Trans. Parallel and Distributed Systems, vol. 1, no. 3, pp. 330-14, Jul. 1990.

[20]. T. Yang and A. Gerasoulis, "List Scheduling with and without Communication Delays," Parallel Computing, vol. 19, pp. 1321-24, 1993.

[21]. YK. Kwok and I. Ahmad, "Dynamic Critical-Path Scheduling: An Effective Technique for Allocating Task Graphs to Multiprocessors," IEEE Trans. Parallel and Distributed Systems, vol. 7, no. 5, pp. 506-16, May 1996.

[22]. TL. Adam, KM. Chandy and J. Dickson, "A Comparison of List Scheduling for Parallel Processing Systems," Comm. ACM, vol. 17, no. 12, pp. 685-16, Dec. 1974.

[23]. C. McCreary and H. Gill, "Automatic Determination of Grain Size for Efficient Parallel Processing," Comm. ACM, vol. 32, pp. 1073-6, Sep. 1989.

[24]. J. Baxter and JH. Patel, "The LAST Algorithm: A Heuristic-Based Static Task Allocation Algorithm.," In: Proceeding of the 1989 Int'l Conf. Parallel Processing, vol. II, pp. 217-6, Aug. 1989.

[25]. JJ. Hwang and *et al.*, "Scheduling Precedence Graphs in Systems with Interprocessor Communication Times," SIAM J. Computing, vol. 18, no. 2, pp. 244-14, 1989.

[26]. GC. Sih and EA. Lee, "A Compile-Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processor Architectures," IEEE Trans. Parallel and Distributed Systems, vol. 4, no. 2, pp. 75-13, 1993.

[27]. R. Hwang, M. Gen and H. A. Katayama, "Comparison of multiprocessor task scheduling algorithms with communication costs," Computer & Operations Research, vol. 35, pp. 976-18, 2008.

[28]. M. Dorigo, G. DiCaro and L. Gambardella, "Ant Algorithm for Discrete Optimization," Artificial Life, vol. 5, no. 2, pp. 137-36, 1999.

[29]. S. Wolfram, "Cellular Automata," Los Alamos Science, vol. 9, pp. 2-21, 1983.

[30]. K. S. Narendra and M. A. L. Thathachar, "Learning Automata: A Survey," IEEE Trans. Syst., Man, Cybern., vol. SMC-14, pp. 323–12, 1974.

[31]. M. A. L. Thathachar and P. S. Sastry, "Varieties of Learning Automata: An Overview," IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 32, no. 6, pp. 711–12, Dec. 2002.

[32]. M. R. Meybodi, H. Beigy and M. Taherkhani, "Cellular Learning Automata and Its Applications," Journal of Science and Technology of Sharif University, vol. 25, pp. 54-24, 2004.

[33]. H. Beigy and M. R. Meybodi, "Cellular Learning Automata With Multiple Learning Automata in Each Cell and Its Applications," IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 40, no. 1, pp. 54-12, Feb. 2010.

[34]. M. Asnaashari and M. R. Meybodi, "Irregular Cellular Learning Automata and Its Application to Clustering in Cellular Networks," In: Proc. 15th Conf. on Electrical Engineering, vol. on Communication, pp. 14-15, Tehran, May 2007.

[35]. MA. Al-Mouhamed, "Lower Bound on the Number of Processors and Time for Scheduling Precedence Graphs with Communication Costs," IEEE Trans. Software Engineering, vol. 16, no. 12, pp. 1390-12, Dec. 1990.

[36]. A. Al-Maasarani, *Priority-Based Scheduling and Evaluation of Precedence Graphs with Communication Times*, M.S. Thesis, King Fahd University of Petroleum and Minerals, Saudi Arabia, 1993.

[37]. H. R. Boveiri, "Task Assigning Techniques for List-Scheduling in Homogeneous Multiprocessor Environments: A Survey," International Journal of Software Engineering and Its Applications (IJSEIA), vol. 9, no. 12, pp. 303-10, Dec. 2015.

[38]. H. R. Boveiri, "An Efficient Task Priority Measurement for List-Scheduling in Multiprocessor Environments," International Journal of Software Engineering and Its Applications (IJSEIA), vol. 9, no. 5, pp. 233-14, May 2015.

An Efficient Hybrid Approach Based on the ACO and CLA for Static Task-Graph Scheduling in Homogeneous Multiprocessor Environments

Hamid Reza Boveiri*
* Corresponding Author: Sama Technical and Vocational College, Islamic Azad University, Shoushtar Branch, Shoushtar, Iran.
boveiri@{shoushtar-samacollege.ir or ieee.org}

Abstract- Task scheduling has been so far of important challenges in high-performance computers e.g. parallel and distributed systems. Using such architectures during compiling, each application program is divided to some tasks. Because of data-flow among the tasks, they may be dependent to one another; hence, there will be precedence constraints and communication delays among them so that each application with its corresponding tasks can be modeled using a Directed Acyclic Graph (DAG) named task graph. In static task-graph scheduling in homogeneous multiprocessor environments, tasks in the given task graph should be mapped to a predefined number of identical processing elements regarding the precedence constraints and communication delays so that the program's completion time (finish time) is minimized, and this is an NP-hard problem form the time-complexity perspective. Actually, the achieved results are dominated by two different-in-nature factors: 1) which topological order of tasks should be considered? (sequencing subproblem), and 2) how should the extracted order be distributed over the processors? (assigning subproblem). In this paper, an efficient hybrid approach is proposed in which the Ant Colony Optimization (ACO) determines the order of tasks, and a Cellular Learning Automata (CLA) machine tackles with the assigning subproblem, and maps the task order derived by ACO to the existing processors. 125 randomly-generated task graphs with different shape parameters such as size, Communication-to-Computation Ratio (CCR), and parallelism are used for the comparison study, and the results shows that the proposed approach is more successful than the traditional counterparts from the performance point of view, and eventually outperforms them.

Keywords- Ant colony optimization (ACO); cellular learning automata (CLA); meta-heuristics; multiprocessor task-graph scheduling; parallel and distributed systems.