بسم الله الرحمن الرحيم

وزارت علوم، تحقیقات و فناوری

موسسه آموزش عالی سراج

دانشکده فنی و مهندسی – گروه مهندسی برق و کامپیوتر

پایان نامه برای دریافت درجه کارشناسی ارشد «M.Sc.»

گرایش : تکنولوژی نرم‌افزار

عنوان :

زمانبندی کار در سیستم‌های ناهمگن با استفاده از الگوریتم شبیه‌سازی تبرید

استاد راهنما :

دکتر سعید تقوی افشرد

استاد مشاور :

دکتر بهمن آراسته

نگارش :

مهدی حیدری طرزم

تابستان ۱۳۹۷

چکیده فارسی: زمانبندی بهینه کارها در سیستم‌های محاسباتی ناهمگن از مسایل پر اهمیت و یک مساله NP complete در زمینه پردازش با کارایی بالا قرار دارد. برنامه‌ها، توالی‌هایی از وظایف در نظر گرفته می‌شوند که با استفاده از گراف جهت‌دار بدون دور قابل نمایش هستند. هر وظیفه به عنوان یک گره از گراف دارای زمان اجرای جداگانه برای هر پردازنده است. به همین صورت هرلبه گراف نمایانگر هزینه انتقال در صورت اجرا در پردازنده غیر مقیم است. در این پژوهش به ارایه الگوریتم پیشنهادی با استفاده از تکنیک شبیه‌سازی تبرید برای زمانبندی وظایف نمایش داده شده در گراف می‌پردازیم که با استفاده از آن می‌توان طول زمان اجرای زمانبندی‌ها را کاهش داد. الگوریتم پیشنهادی شامل دو بخش اصلی می‌شود: الف) ورودی اولیه که در آن یک زمانبندی قابل اجرا و نزدیک به بهینه با پیچیدگی زمانی کم را تولید می‌شود. ب) بخش بهبود زمانبندی که در آن زمانبندی تولید شده در مرحله اول با استفاده از تکنیک شبیه‌سازی تبرید بهبود داده می‌شود. تمرکز الگوریتم در هر فاز پیدا کردن زمانبندی بهینه با طول زمان اجرای کمینه است. الگوریتم پیشنهادی با استفاده از گراف‌های بدون دور جهت دار آزمایش و نتایج آن با الگوریتم‌های شناخته شده و دارای زمانبندی‌های کمینه مقیسه شده است. نتایج نشانگر این است که در طول زمانبندی‌ها تا ۲ ٪ بهبود به وجود آمده است.

واژگان کلیدی : زمانبندی کار – سیستم‌های ناهمگن – شبیه سازی تبرید – گراف بدون دور جهت‌دار

فهرست مطالب

فهرست جداول

فهرست شکل‌ها

فصل ۱: کلیات تحقیق

۱-۱ مقدمه

سالهای اخیر شاهد رشد چشمگیری در محبوبیت استفاده از سیستم‌های محاسبات ابری و خدمات مربوط به آن بوده‌ایم. از آنجایی که رشد و ظهور محاسبات موازی بسیار امیدوار کننده می‌باشد، لذا خدمات و سرویس های مربوط به محاسبات ابری به عنوان منبع اولیه توان محاسباتی مشاغل، افراد و نرم افزارهای محاسباتی سیار محسوب می‌گردد. ارائه کنندگان خدمات مربوط به محاسبات ابری از یک پایگاه مشترک داده‌ای بهره گرفته و از یک سرور مشترک جهت ارائه خدمات به مشترکان خود استفاده نموده که براساس میزان تقاضای خود به راحتی به این پایگاه دسترسی پیدا کرده، ارتباط ایجاد نموده و منابع خود را بصورت محاسبات طبقه‌بندی شده و مقیاس‌پذیری ذخیره‌سازی می‌نمایند. محیط محاسبات ابری در حالت کلی ناهمگن بوده و ماهیت اصلی این محیط ناهمگن که به طور معمول پردازنده‌های چند هسته‌ای مجهز به شتاب دهنده‌های اختصاصی مانند GPU با سرعت بالا با قابلیت اتصال داشته و می‌توانند جهت انجام برنامه‌های مختلف محاسباتی فشرده که این نوع از محاسبات نیز دارای انواع متفاوتی بوده، ارتباط پیدا نمایند. سیستم توزیع‌شده ناهمگن نوعی سیستم حاوی گروهی از پردازنده‌های متنوع بوده که از طریق شبکه سریع به یکدیگر متصل شده‌اند. این سیستم جهت اجرای موازی و همزمان برنامه‌های توزیعی در نظر گرفته شده و مورد استفاده قرار می‌گیرد. به عبارتی از این سیستم جهت اجرای موازی برنامه توزیع شده می‌توان استفاده نمود. همانطور که رقابت ارائه کنندگان خدمات و سرویس‌های ابری روز به روز در حال افزایش می‌باشد و هزینه‌ها یا به عبارتی قیمت سرویس‌ها کاهش می‌یابد، لذا ارائه کنندگان خدمات و سرویس‌ها بایستی هزینه‌های خود را به حداقل رسانده و کاهش دهند تا میزان درآمد ناشی از ارائه خدمات افزایش یابد که این امر عمدتاً با افزایش استفاده از منابع سخت افزاری حاصل می‌گردد. برنامه‌ریزی زمانبندی وظایف یا دستورات شامل فرایندی بوده که در آن نقشه‌ای برداری از وظایف یا دستورات که باید پردازش شوند تهیه شده و هر یک از عملکردها ثبت می‌گردد که این امر نقش بسیاری حیاتی در استفاده از منابع ایفا نموده و در

بهره‌برداری از منابع بسیار ضروری می‌باشد. بنابراین، تمایلات جدیدی بر روی برنامه‌ریزی دستورات یا وظایف برای سیستم های محاسباتی ناهمگن وجود خواهد داشت[۱۹].

۱-۲ تعریف مساله

روش‌های مربوط به برنامه‌ریزی و زمانبندی دستورات و یا وظایف در سیستم‌های چند پردازشی همگن، که از جمله‌ی آن می‌توان به توپولوژی شبکه‌ای و پیکربندی توپولوژی شبکه اشاره نمود که از آن در جهت حل مسائل محاسباتی پردازش تصویری متمرکز و نرم‌افزارهای تصویری بهره گرفته می‌شود[۱۸-۵]. با این حال، ناهمگونی در سیستم‌های محاسباتی باعث افزایش میزان پیچیدگی مسائل مربوط به زمانبندی و برنامه‌ریزی گردیده و میزان پیچیدگی این مسائل نسبت به سیستم های چند پردازشی همگن بیشتر خواهد بود[۱۹]. هنگام تخصیص هر یک از فعالیت ها و دستورات به پردازنده، روش برنامه‌ریزی و تکنیک زمانبندی بایستی به سرعت پردازنده‌ها و تنوع سرعت آنها توجه داشته باشیم. جهت استفاده بهینه از سیستم‌های محاسباتی ناهمگن، می‌توان برنامه کاربردی را به مجموعه‌ای از دستورات و یا وظایف (بخش برنامه) تقسیم نموده که این امر با استفاده از یک نمودار اجرایی تصادفی و لبه‌های وزنی نمودار DAG صورت گرفته که در آن هر یک از دستورات و یا وظایف از لحاظ محاسباتی همگن و یک‌دست بوده و بر این اساس می تواند به بهترین وجه ممکن به مناسب‌ترین پردازنده اختصاص داده شود. عملکرد یک برنامه موازی در پردازنده‌های ناهمگن به شدت به ویژگی‌های برنامه بستگی داشته (زمان اجرایی، داده‌ها و ارتباط بین دستورات) و نیز سطح ویژگی‌ها نیز اثر بسزایی خواهد داشت (برای مثال ظرفیت محاسباتی پردازنده ها، تعداد پردازنده‌ها، پهنای باند ارتباطی، اندازه حافظه و غیره). به منظور استفاده موثر و بهینه از سیستم های محاسباتی ناهمگن، بایستی طول برنامه را به حداقل برسانیم و به عبارتی برنامه زمانی یا فهرست مربوط به آن به حداقل میزان ممکن رسید. همچنین در کنار این امر و علاوه بر به حداقل رساندن فهرست زمانی و برنامه‌ریزی بایستی سعی بر به حداکثر رساندن توان سیستم و افزایش بهره وری پردازنده جهت کاهش هزینه ها نیز نمائیم[۱۳-۶-۱۹].

الگوریتم مربوط به زمانبندی با اختصاص فعالیت و عملکرد پردازنده موجود می‌تواند باعث تسهیل فعالیت پردازنده گردیده و عملکرد آنرا تکمیل نماید، لذا از این رو بازدهی و راندمان آن افزایش پیدا خواهد کرد. زمانبندی کار در سیستم‌های ناهمگن یک مساله NP کامل است[۱۷-۳-۱۹]. به طور عمده مسائل مربوط به وظایف یا عملکرد زمانبندی به دو دسته برنامه‌ریزی استاتیک و برنامه‌ریزی داینامیک (ایستا و پویا) اصلی تقسیم می‌گردد[۱۴-۴]. در برنامه‌ریزی ایستا، تمامی اطلاعات موجود در مورد وظایف و عملکرد مانند زمان اجرا و هزینه ارتباط، از قبل تعیین گردیده و مشخص می‌باشد. از طرف دیگر، در برنامه ریزی پویا، زمان اجرا و تعداد دفعات ارتباطی در زمان اجرا تعیین خواهد شد. پس از اجرای هر دستور، مقادیر جدید سطح بعدی محاسبه خواهد شد. الگوریتم‌های زمانبندی استاتیک به دو گروه اصلی تقسیم می گردد: روش‌های مبتنی بر لیست و روش‌های مبتنی بر جستجوی تصادفی این دو روش محسوب می گردند. الگوریتم مبتنی بر لیست نتایج را با پیچیدگی زمانی قابل قبول ارایه می‌کنند ولی این نتایج به قدر کافی بهینه نیستند. در برابر این امر، الگوریتم‌های مبتنی بر جستجوی تصادفی راه حل بسیار مطلوب‌تری را ارایه می‌کنند ولی پیچیدگی زمانی بسیار بالایی را به همراه دارند. به طور کلی نسخه جستجوی تصادفی نزدیک‌ترین راه‌حل به راه‌حل بهینه را ارائه می‌کنند ولی یکی از عمده موانع و مشکلات پیش روی الگوریتم‌های جدید و نوین تعداد تکرار این الگوریتم‌ها است. این باعث شده تعداد بسیار زیادی از الگوریتم ها جهت حصول بهترین راه حل ممکن عمل جستجو را تکرار کنند که این امر با حالت بهینه و مطلوب الگوریتم‌های موجود متناقض می‌باشد.

هدف اصلی الگوریتم‌های مربوط به زمانبندی ایجاد تعادل بین ارتباطات ایجاد شده و همزمان‌سازی فعالیت‌ها می‌باشد به عبارتی در راستای موازی‌سازی فعالیت‌های موجود قدم بر می‌دارد. در یک فضای جستجوی مناسب، نمونه‌های موجود به راه‌حل و روش بهینه نزدیک می‌باشند. بنابراین ما بایستی جمعیت یا نقطه شروع اولیه را طوری طراحی نموده و ایجاد نمائیم که در زمان اجرای مناسبی به بهترین راه‌حل و روش ممکن دست یابیم.

در این پژوهش، روشی برپایه جستجوی تصادفی ارایه می‌کنیم که برای کاهش زمان جستجو، فهرست زمانبندی حاصل از یک روش استخراج اکتشافی را به عنوان جمعیت اولیه در نظر گرفته و سپس با استفاده از شبیه‌سازی تبرید، نتایج حاصل شده را بهینه می‌کنیم. الگوریتم پیشنهادی باعث ایجاد نوعی الگوریتم جدیدی گردیده که در آن جمعیت اولیه در زمان کم یک زمانبندی نزدیک به زمانبندی بهینه است. این زمانبندی نزدیک به بهینه باعث افزایش کیفیت زمانبندی‌ها شده و ترکیب آن با دیگر زمانبندی‌ها باعث ترویج تنوع در بین نمونه‌هایی می‌گردد که در هر یک از تکرارها ایجاد خواهد شد و به عبارتی این اپراتور باعث تنوع در جمعیت خواهد شد. با استفاده از این روش ترکیبی جدید تعداد برنامه‌های تکراری ایجاد شده در جمعیت موجود به حداقل میزان خود خواهد رسید.

فهرست زمان بندی یا لیست برنامه‌ریزی متشکل از دو مرحله اصلی بوده که شامل اولویت بندی دستور و اختصاص وظیفه یا فعالیت (دستور) به پردازنده انتخابی می باشد. در مرحله اولویت بندی کار، هر یک از وظایف براساس اولویت تعیین گردیده و لذا اهمیت آن در کل برنامه تعیین می‌گردد. در ابتدا فهرستی از تمام وظایف و دستوراتی که هیچگونه پیش پردازشی نداشته‌اند تهیه می گردد. یکی از دستورات یا وظایفی که بالاترین اولویت را داشته انتخاب گردیده و براساس مکانیسم انتخاب پردازنده به پردازنده موجود اختصاص داده می شود[۲].

سپس زمانبندی به دست آمده به عنوان جمعیت اولیه به همراه چند زمانبندی قابل اجرا دیگر به الگوریتم شبیه‌سازی تبرید داده می‌شود. در هر مرحله همسایه‌های جدید با ترکیب کردن جمعیت باقی مانده از مرحله قبل تولید می‌گردد. در این مرحله علاوه بر ترکیب ممکن است روی هر یک از نتایج عملی شبیه به عمل جهش در ژنتیک صورت گیرد که این عمل باعث می‌شود جستجو در فضای بیشتری انجام شود. در نهایت بعد از تکرار چند باره این عمل و رسیدن به دمای مورد نظر که شرط اتمام کار است، الگوریتم زمانبندی بهینه‌ای را که دارای زمان اجرای کمتری است را ارایه می‌کند.

۱-۳ فرموله سازی مساله

در ادامه این بخش، سعی بر این شده تا با فرموله‌سازی مسائل مربوط به زمانبندی فعالیت و دستورات غیرانحصاری استاتیک را در سیستم‌های محاسباتی ناهمگن ارائه نمائیم که در این بین می توان به سیستم های CPU-GPU با هدف به حداقل رساندن زمان کلی اجرا اشاره نمود. یک برنامه کاربردی، متشکل از مجموعه‌ای از فعالیت‌ها و یا وظایف بوده که نمایش آن توسط یک گراف جهت‌دار بدون دور (DAG) صورت می‌گیرد. در این نمودار G=(T<E) بوده که در آن $T = \{t_i, t_i = 1,...,n\}$ در مجموعه‌ای از n فعالیت یا دستور می‌باشد. علامت < نشان دهنده دستور جزئی (وظایف یا عملکرد کوچک) در T می‌باشد. در هر دو فعالیت و دستور موجود (t_i, t_j) حالت $\in T$ را خواهیم داشت و به عبارتی $(t_i, t_j) \in T$ خواهد بود که نشانگر دستور جزئی بوده و به عبارتی $t_i < t_j$ می‌باشد که این امر بدین معنی می‌باشد که فعالیت یا دستور t_j تا زمانی که دستور یا فعالیت t_i تکمیل نگردیده باشد زمان‌بندی نگردیده و برنامه‌ریزی نخواهد شد چرا که از خروجی محاسبات t_i در t_j استفاده خواهد شد. از این رو، t_i یک پیش پردازه بوده و به عنوان پیش پردازه t_j است و t_j نیز جانشین t_i می‌باشد. E نیز مجموعه‌ای از لبه‌های مستقیم و یا قوس و کمان‌ها می‌باشد. A شامل مقدار C_{ij} اختصاص داده شده به هر یک از قوس‌هایی بوده که مقداری از داده‌ها و اطلاعات را جهت جابجایی و انتقال از دستور یا فعالیت t_i به فعالیت t_j در واحد بایت را شامل می‌گردد[۸].

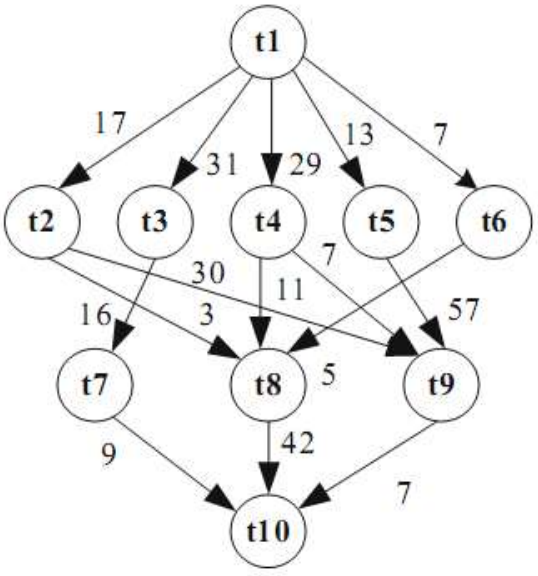
یک سیستم محاسباتی ناهمگن حاوی مجموعه‌ای از پردازنده های $P = \{P_j : j = 0,...,m-1\}$ بوده که با استفاده از یک توپولوژی کامل متصل شده است. در محاسبه ECT (محاسبه زمان تخمینی) در یک دستور یا وظیفه، اگر t_i و t_j به یک پردازنده مشترک اختصاص داده شود، معادل صفر خواهد بود. در غیر اینصورت، داده‌ها و اطلاعات از پردازنده‌ای انتقال یافته که در آن دستور t_i به پردازنده اختصاص داده شده که در آن دستور t_j اختصاص داده شده و قرار گرفته است[۴].

تعدادی از اصطلاحات مورد استفاده در این تحقیق به شرح زیر می‌باشند:

$Level(n_i), pred(n_i), succ(n_i), makespan, EST(n_i P_j)$ ، و $EET(n_i, P_j)$ که بیانگر زمان اتمام پردازش در گره n_i در DAG می باشد. گره ورودی شامل گرهی بوده که هیچ پیش پرازنده‌ای ندارد. $succ(n_i)$ نشان دهنده پردازش سریع در دستور n_i می‌باشد. گره خروجی هیچ‌گونه جایگزینی ندارد. توجه داشته باشید که t_i و n_i می-توانند جایگزین یکدیگر بوده و متناوباً به جای یکدیگر در دستورات استفاده گردند. علاوه بر این، Makespan نیز به طول کل اجرای زمانبندی یا برنامه‌ریزی اشاره دارد. این امر بدین معنی می باشد که makespan معادل AFT (زمان واقعی پایانی) در گره خروجی DAG می‌باشد. بنابراین، makespan بصورت زیر تعریف می‌گردد:

$$makespan = \max\{AFT\ (nexit)\}$$

(۱-۱)



Task	P1	P2
t1	171	125
t2	133	114
t3	26	131
t4	145	192
t5	120	184
t6	10	152
t7	114	30
t8	50	126
t9	191	65
t10	3	2

شکل ۱-۱- مثال DAG و زمان تقریبی محاسبات، w_i, j فعالیت و دستورات در هر پردازنده

حداکثر تعداد لبه‌های بین یک گره و گره ورودی بیانگر سطح فعالیت یا کار در DAG می‌باشد. در گره ورودی میزان (n_{entry}) معادل صفر خواهد بود. علاوه بر این، طولانی‌ترین مسیر بین گره ورودی و خروجی به عنوان مسیر اصلی DAG شناخته شده است. اولین زمان شروع گره n_i در پردازنده P_j بصورت زیر نشان داده می شود:

$$EST\,(ni, pj) = \max\left\{Tavailable\,(pj)\,,\ \max_{nm\in pred(ni)}\{AFT\,(nm) + cm, i\}\right\}$$

(۲-۱)

که در آن $T_{available}(P_j)$ نشانگر زمان بوده که در این زمان پردازنده P_j آماده اجرای فعالیت یا وظیفه جدیدی می باشد. علاوه بر این، حداکثر بلوک‌های داخلی نشان دهنده مجموع زمان مورد نیاز جهت انتقال تمام داده می‌باشد. در گره ورودی، از آنجایی که تمام پردازنده‌ها آماده اجرا بوده و هیچ گونه پشتیبانی وجود ندارد، در این صورت $EST(n_{entry},P_j)$ معادل صفر خواهد بود.

از طرف دیگر، $EFT(n_i,P_j)$ زمان اولیه پایانی فعالیت یا دستور n_i در پردازنده P_j را نشان می‌دهد. علاوه بر این، EFT مجموع کلی EST پردازنده را در بر گرفته و نیز شامل مقادیر محاسباتی فعالیت در همان پردازنده می-باشد. بنابراین، $EFT\ (n_i,P_j)$ به شرح زیر تعریف می‌گردد.

$$EFT\,(ni, pj) = EST\,(ni, pj) + wi, j \qquad (۳-۱)$$

چنین فرض می‌گردد که DAG ورودی دارای گره ورودی منفرد و یا واحدی به همراه گره خروجی بوده و به عبارتی دارای یک گره ورودی و یک گره خروجی می‌باشد. بنابراین، اگر DAG دارای بیش از یک گره ورود یا خروجی باشد، در اینصورت دو گره به ترتیب به گره اصلی و گره ثانویه یا فرزند متصل خواهد شد. بطور کلی، مسائل مربوط به زمانبندی جهت تخصیص فعالیت و عملکردها برای اجرای دستورات و فعالیت هر چه بهتر پردازنده‌ها مورد استفاده قرار می‌گیرد که در این مسائل تمام محدودیت‌های مذکور در نظر گرفته شده و از طرفی نیز زمان اجرای کلی نیز به حداقل خواهد رسید.

۱-۲ ساختار پایان‌نامه

در ادامه، تحقیق بصورت زیر ارائه گردیده است: در بخش ۲ تمامی فعالیت های مربوط به فهرست زمانبندی یا لیست برنامه ریزی در سیستم های محاسباتی ناهمگن ارائه گردیده است. در بخش ۳، الگوریتم پیشنهادی ارائه گردیده و نمونه و مثالهای مربوط به آن نیز به همراه مقایسه نتایج حاصله با نتایج یافته‌های قبلی که پیش از این گزارش گردیده ارائه شده است. در بخش ۴ نتایج تجربی الگوریتم ها مورد بحث و بررسی قرار گرفته شده است. در نهایت نتیجه گیری و برخی فعالیت های آتی در بخش ۵ ارائه شده است.

فصل۲: مبانی نظری و پیشینه تحقیق

الگوریتم‌های زمانبندی فعالیت‌ها و برنامه‌ریزی وظایف در سیستم‌های همگن به خوبی توسط محققان مورد بررسی قرار گرفته است. در این بخش، سعی بر این شده تا فعالیت های صورت گرفته شده به‌طور خلاصه مطرح گردیده و الگوریتم‌های پیشنهادی قبلی با توجه به راه‌حل‌ها و روش‌های مطلوب و نسبتاً موثر به صورت خلاصه مطرح گردد.

الگوریتم مربوط به زمانبندی استاتیک بعنوان الگوریتم‌های جدید شناخته شده و در گروه‌های الگوریتم‌های جدید و متا طبقه‌بندی می‌گردد. الگوریتم‌های جدید در واقع به مدل و روش هایی براساس لیست، خوشه بندی (گروه بندی) و متدها تکثیر و نسخه برداری تقسیم می‌گردد.

در روش‌های جدید و مبتنی بر لیست، تناسب براساس عملکرد هر یک از فعالیت‌های موجود صورت گرفته و براساس اولویت عملکردهای موجود لیست بندی صورت می‌گیرد. در این روش اکتشافی، انتخاب فعالیت و عملکرد جهت پردازش براساس اولویت انجام گرفته و لذا هر یک از فعالیت هایی که تناسب بهتری داشته باشد زودتر پردازش خواهد شد. اولین زمان مربوط به پایان نامتناجس یا ناهمگون [۱۶](HEFT)، مسیر اصلی در پردازنده (PEFT) و الگوریتم ارایه شده توسط Imtiaz Ahmad و همکارش[۲] نمونه‌های مطرحی از روش های مربوط به لیست‌بندی می‌باشد.

۲-۲ پیشینه تحقیق

HEFT (زمان پایانی اولیه ناهمگن) یکی از بهترین فرضیه‌هایی بوده که توسط Topcuoglu و همکارانش مطرح گردید. الگوریتم HEFT جهت تعداد پردازنده‌های ناهمگن طراحی گردیده و در نظر گرفته شده که در برگیرنده دو مرحله می باشد: در مرحله اول تناسب زمانبندی محاسبه گردیده و در مرحله دوم انتخاب پردازنده صورت گرفته و همچنین آنالیز براساس تناسب با پردازنده انجام یافته که این امر باعث می گردد تا در کمترین زمان ممکن پردازش صورت گرفته و زمان پایان فرایند به حداقل زمان ممکن رسد. در این الگوریتم، تناسب براساس جفت پارامترها مشخص گردیده که این پارامترها با عنوان t_{level} و b_{level} شناخته شده‌اند. در نمودار مربوط

به فعالیت‌ها و عملکردهای موجود، t_{level} (یا سطح t) نشانگر طولانی‌ترین مسیر ممکن از گره آغازی تا گره مورد نظر می باشد. از دیدگاه دیگر، t_{level} هر یک از گره‌ها بیانگر نزدیکترین زمان شروع فرایندی بوده که این گره آغاز نموده و b_{level} هر گره نیز بیانگر طولانی‌ترین مسیر جهت خروج گره می‌باشد.

در اکثر فعالیت های قبلی HEFT با سایر روش های اکتشافی مقایسه گردیده و به این نتیجه رسیدند که این روش می تواند موثرترین برنامه را با حداقل زمان اجرایی یا makespan در نظر گرفته شود[۱۱-۱۰-۴-۱۵]. در این روش پیچیدگی زمانی $O(t^2P)$ بوده که در آن t نشان دهنده تعداد فعالیت یا وظایف اجرایی بوده و P تعداد پردازنده ای که در اجرای فعالیت نقش دارد را نشان خواهد داد. شایان ذکر است که $O(t^2P)$ کمترین پیچیدگی زمانی برای ارایه زمانبندی قابل قبول نسبت به زمان به دست آوردن آن می‌باشد.

فعالیت و عملکرد فاز رتبه‌بندی بطور معمول براساس میزان فعالیت و نیز حجم ارتباطات بین فعالیت ها تعیین می‌گردد. الگوریتم HEFT به شرح زیر تعریف می‌گردد:

$$rankHEFT(ni) = \overline{wi} + \max_{nj \in succ(ni)} \left\{ \overline{ci,j} + rankHEFT(nj) \right\}$$

(۱-۲)

تابع رتبه بندی در $rankHEFT(n_i)$ بیانگر طولانی ترین مسیر بین گره موجود و گره خروجی در DAG که با اتمام اجرای آن تمامی کارها به پایان خواهد رسید می‌باشد. گره خروجی با استفاده از مجموعه‌ای از زمان محاسباتی (میانگین زمان محاسباتی) پردازنده ها محاسبه خواهد شد. با توجه به معادله رتبه‌بندی یکسان می‌توان به این نتیجه رسید که $\overline{w_i}$ بیانگر زمان محاسباتی اجرای فعالیت n_i توسط مجموعه ای از پردازنده‌ها می باشد. علاوه بر این، $C_{i,j}$ بیانگر میزان ارتباط بین فعالیت های n_i و n_j می باشد. در نهایت فهرست عملکردها و یا وظایف سازماندهی شده ارائه گردیده که هر گره اولویت اجرایی بالایی داشته و از لحاظ اجرایی نیز اهمیت دارند.

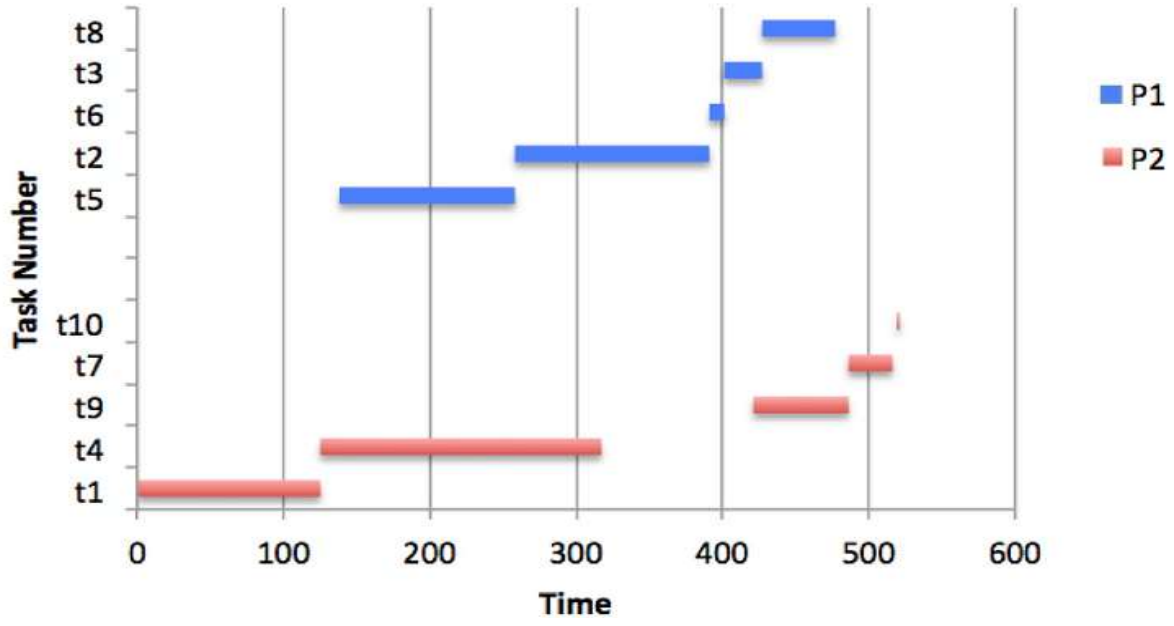
فاز دوم الگوریتم شامل انتخاب پردازنده بوده که در آن لیست مرتب سازی شده به هر یک از پردازنده های اجرایی متصل گردیده و با یکدیگر ارتباط ایجاد می نمایند. پردازنده‌ای که کوتاه‌ترین زمان پایانی فعالیت n_i را

انتخاب می نماید. بنابراین، در صورت امکان، الگوریتم HEFT وظایف و عملکردها را در اولین زمان ممکن بین دو زمان بندی موجود در یک پردازنده قرار خواهد داد.

از آنجایی که می خواهیم نتایج حاصل از بخشی از الگوریتم را با HEFT مقایسه نمائیم، لذا جزئیات مربوط به آن با استفاده از مثالی توضیح داده می شود. با توجه به مثال موجود در شکل ۱، ابتدا مقادیر رتبه‌بندی مربوط به $HEFT(n_i)$ محاسبه گردیده و در جدول ۱ ارائه شده است. الگوریتم HEFT به صورت کاهشی و مرتب سازی از کم به بیشتر براساس اولویت‌بندی $\{t_1, t_5, t_4, t_2, t_6, t_3, t_9, t_8, t_7, t_{10}\}$ انجام خواهد داد. سپس هر یک از فعالیت های موجود در فهرست اولویت‌بندی به یکی از بهترین پردازنده‌ها اختصاص داده شده تا بصورت اجرایی در آمده و اجرا گردد. در الگوریتم HEFT، پردازنده‌ای که وظایف و فعالیت‌ها را با کمترین زمان پایانی یا حداقل زمان پایانی (EFT) انتخاب می‌گردد. برای مثال، چنین فرض می‌گردد که تمامی پردازنده ها موجود می‌باشد. بنابراین، کوتاه ترین زمان آغازی (EST) در هر دو پردازنده موجود در مثال معادل صفر خواهد بود. سپس، EFT برای هر دو پردازنده محاسبه خواهد شد. پردازنده ۱ قادر خواهد بود تا t1 را ۱۷۱ واحد زمانی اجرای نموده در حالی که پردازنده ۲ همان t1 را در ۱۲۵ واحد زمانی اجرا خواهد نمود. بنابراین، از آنجایی که پردازنده ۲ دارای کمترین میزان EFT می باشد، لذا جهت اجرای t1 انتخاب خواهد شد. زمانی که t1 تکمیل می‌گردد، فعالیت بعدی، t5، براساس پردازنده انتخابی شروع به اجرا خواهد نمود. این فرایند تا اتمام آخرین فعالیت ادامه خواهد داشت و تا زمانی که t10 اجرا گردد ادامه پیدا خواهد کرد. همانطور در این مثال، در الگوریتم HEFT، تمام فعالیت ها اجرای خود را تکمیل نموده و در بازه زمانی یا makespan معادل ۵۲۱ واحد زمانی اجرا تکمیل خواهد شد. علاوه بر این، در جدول ۱ جزئیات مربوط به داده‌های زمانبندی ارائه شده در الگوریتم HEFT را نشان می‌دهد. نمودار ۲-۱ نیز جزئیات مربوط به همان مثال را نشان می‌دهد. همانطور که در شکل نیز قابل مشاهده می‌باشد، فعالیت $\{t_1, t_4, t_9, t_7, t_{10}\}$ توسط پردازنده ۲ قابل اجرا بوده در حالی که فعالیت‌های $\{t_5, t_2, t_6, t_3, t_8\}$ در پردازنده دیگری اجرا خواهد شد.

جدول ۲-۱ :برنامه تولید شده توسط الگوریتم HEFT

Tasks	$rank\,HEFT(ni)$	Processor selected	Earliest start time (EST)	Earliest finish time (EFT)
t1	507.5	2	0	125
t5	346.5	1	138	258
t4	313.0	2	125	317
t2	291.0	1	258	391
t6	218.5	1	391	401
t3	178.0	1	401	427
t9	137.5	2	421	486
t8	132.5	1	427	477
t7	83.5	2	486	516
t10	2.5	2	519	521



شکل ۲-۱ –نحوه تخصیص پردازنده در الگوریتم زمان بندیHEFT

یکی دیگر از الگوریتم های مطرح و شناخته شده در زمینه زمانبندی و فهرست زمانبندی شامل الگوریتم PEFT بوده که توسط عرب نژاد و همکارانش[۳] اجرا شده است و دارای پیچیدگی یکسانی باHEFT به میزان $O(t^2p)$ دارد. PEFT پیش نمایش از جدول محاسبات و مقادیر OCT را ارائه می‌نماید. مقادیر ارایه شده در این جدول بسیار بهینه بوده که در این جدول دسترس بودن و یا موجودیت پردازنده در نظر گرفته نمی‌شود. همان‌طور

که در الگوریتم HEFT نیز قابل مشاهده است، PEFT نیز دارای دو فاز اصلی بوده که به شرح زیر می‌باشد. فاز اولویت بندی فعالیت و کار و فاز انتخاب پردازنده و مرحله انتخاب پردازشگر.

جدول ۲-۲-برنامه تولید شده توسط الگوریتمPEFT

Tasks	$rankOCT\ (ni)$	Processor selected	EST	EFT	OEFT
t1	247.5	2	0	125	376
t5	95.5	2	125	309	376
t2	82.0	1	142	275	372
t4	70.5	1	275	420	494
t6	55.5	1	420	430	483
t3	40.0	2	309	440	472
t9	2.5	2	440	505	507
t8	2.5	1	430	480	483
t7	2.5	2	505	535	537
t10	0.0	2	535	537	537

الگوریتم PEFT با محاسبه جدول بهینه فعالیت خود را آغاز نموده که در آن مرتب‌سازی و ترتیب فعالیت‌ها قرار گرفته است. مقادیر OCT مربوط به فعالیت n_i در پردازنده P_k به شرح زیر تعریف می‌گردد:

$$OCT\ (ni,\ pk) = \max_{tj \in succ(ti)} \left[\min_{pw \in P} \left\{ \mathrm{OCT\ (nj,\ p}w) + w\ (nj,\ pw) + \overline{ci,\ j} \right\} \right],$$
$$\overline{ci,\ j} = 0 \quad if\ pw = pk$$

(۲-۲)

خاصیت برگشت پذیری محاسبات OCT باعث می‌گردد تا حداکثر زمان ممکن کوتاه‌ترین مسیر بین فعالیت موجود و گره خروجی محاسبه گردد. تمامی پردازنده‌هایی که در فاز انتخاب پردازنده قرار می‌گیرند بایستی در محاسبات جدول نیز قرار گرفته باشند. در همان معادله، $w(n_j,P_w)$ نشان دهنده زمان اجرای فعالیت n_i در پردازنده Pw می باشد. علاوه بر این، Ci,j بیانگر مجموع میانگین مقادیر ارتباطی بین فعالیت ها می‌باشد. همانطور که در گره خروجی نیز قابل مشاهده می باشد، $OCT(n_{exit},P_w)$ برای تمامی پردازنده‌ها خواهد بود. پس از محاسبه تمام مقادیر OCT در بین تمام ارتباطات بین فعالیت ها و پردازنده ها، می توان پس از آن مرتب‌سازی فعالیت‌ها را

محاسبه نمود. در PEFT، مرتب‌سازی فعالیت‌ها با استفاده از میانگین OCT در هر یک از فعالیت‌های تعریف شده در فرمول زیر قابل محاسبه خواهد بود.

$$rank_{oct}\,(ni) = \frac{\sum_{k=1}^{P} OCT\,(ni, pk)}{P}$$

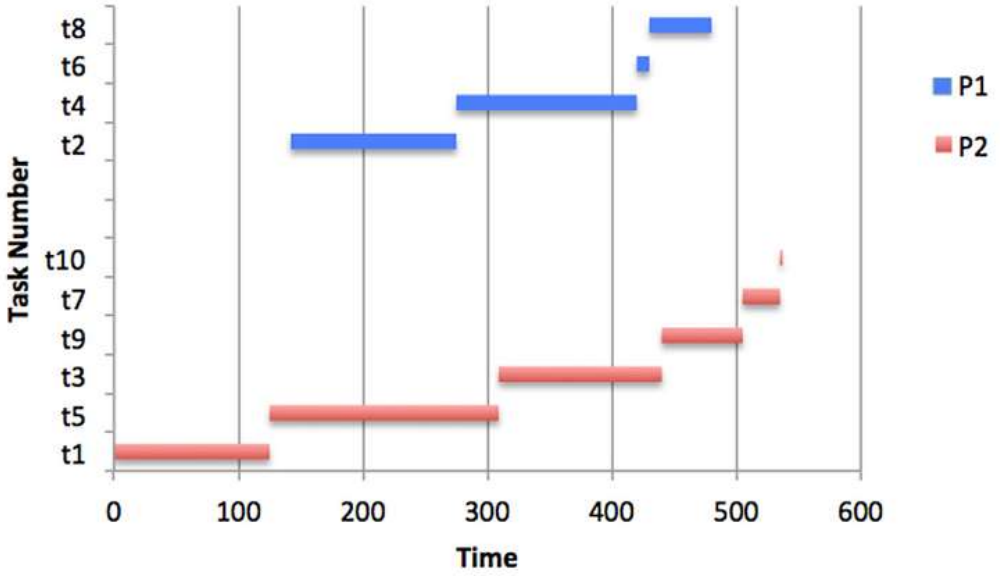
(۳-۲)

با توجه به مثال موجود در شکل ۱-۱ rank OCT جهت به دست آوردن مقادیر جدول ۲-۲ محاسبه خواهد شد. PEFT بصورت کاهشی از کم به زیاد اولویت بندی را براساس فهرست مربوط به مرتب‌سازی $\{t_1, t_5, t_2, t_4, t_6, t_3, t_9, t_8, t_{10}\}$ انجام خواهد داد. در مقایسه با HEFT فعالیت‌های t_2 و t_4 ترتیب خود را جایگزین نموده اند.

در فاز دوم PEFT، که فاز انتخاب پردازنده می‌باشد، مقادیر بهینه EFT (OEFT) محاسبه خواهد شد. تمام جزئیات مربوط به آن در جدول ۲-۲ ارائه شده است. OEFT شامل مجموع EFT بوده زمان محاسبه طولانی‌ترین مسیر از گره موجود تا گره خروجی را در بر می‌گیرد. بنابراین، به جای انتخاب پردازنده‌ای که زمان اولیه کوتاهی داشته، با نگاهی به روش های قبلی می توان کوتاه ترین زمان پایانی را برای فعالیت موردنظر انتخاب نمود. بنابراین OEFT به شرح زیر تعریف می‌گردد:

$$OEFT\,(ni, pj) = EFT\,(ni, pj) + OCT\,(ni, pj)$$

(۴-۲)



شکل ۲-۲ - مثال الگوریتم زمانبندی PEFT

با توجه به الگوریتم PEFT جهت انجام زمانبندی فعالیت ها که در شکل ۱-۱ مورد استفاده قرار گرفت، بازه زمانی ۵۳۷ واحد زمانی حاصل گردید. شکل ۲-۲ جزئیات مربوط به زمان‌بندی را در همان نمونه نشان می‌دهد. همانطور که در شکل ۲-۲ نیز قابل مشاهده می باشد، زمانبندی فعالیت های مربوط به همان مثال و نمونه به دنبال استفاده از الگوریتم PEFT باعث می گردد تا فعالیت $\{t_1, t_5, t_3, t_9, t_{10}\}$ در پردازنده ۲ اجرا گردیده در حالی که فعالیت $\{t_2, t_4, t_6, t_8\}$ در پردازنده ۱ انجام خواهد یافت. بنابراین، مجموع بازه زمانی الگوریتم زمانبندی PEFT، ۵۳۷ واحد زمانی بود.

Hagras و همکارانش[۹] الگوریتم HCPT را در نظر گرفته که در آن فهرست (L) به جای فعالیت مربوط به اولویت‌بندی استفاده گردید. گروه‌های HCPT به مجموعه ای از جفت درختچه های خارج از لیست یا فهرست اختصاص دارد. گره اصلی به عنوان گرم تعریف گردیده که معادل AEST بوده و با ALST یکسان بود. بنابراین، الگوریم با اجرای دو فاز فعالیت نمود. فاز اول شامل فهرست بندی فعالیت‌ها بوده و فاز دوم نیز اختصاص فعالیت به پردازنده بوده که این پردازنده می‌بایستی اجرا می‌نمود. در فاز فهرست گذاری، HCPT فعالیت خود را با فهرست خالی شروع نموده (L) و علاوه بر فهرست خالی دارای یک فهرست توده‌ای نیز بوده (S) که در این فهرست توده‌ای

گره‌های اصلی به ترتیب ALST قرار داده شده بودند. سپس، اگر در گره بالایی یا فوقانی در S گره ریشه‌ای نیز وجود داشته باشد و این گره ریشه‌ای در فهرست وجود نداشته باشد، در اینصورت این گره ریشه‌ای بایستی اجرا گردد. در فاز دوم، فعالیت‌ها و موارد اجرایی به پردازنده‌ای اختصاص داده می‌شود که کوتاه‌ترین و ابتدایی‌ترین زمان اجرایی را داشته باشد.

IILavarsan و همکارانش[۱۲] از الگوریتم HPS استفاده نموده و این نوع از الگوریتم را ارائه نمودند. HPS شامل ۳ فاز اصلی بود. مرتب سازی سطح، اولویت بندی فعالیت یا موارد اجرایی و انتخاب پردازنده، در فاز اول، نرم افزار ورودی DAG به صورت بالا تا پائین سطح به سطح عبور نموده و قرار می‌گیرد. هر یک از فعالیت ها و مواردی که نسبت به یکدیگر مستقل می باشند، به گروه‌هایی متفاوتی اختصاص داده خواهند شد. بنابراین، بایستی توجه داشته باشیم که فعالیت و وظایف بصورت موازی در همان DAG یکسان اجرا گردد. در فاز دوم، اولویت بندی براساس مقادیر DLC صورت گرفته و برای مرتب‌سازی می توان از UCC، LC و DLC بهره گرفت. DLC بیشترین مقدار ارتباط بین تمام جفت گره‌ها را نشان می‌دهد. ULC نیز بالاترین میزان ارتباطی را بین تمام گره‌های کوچک‌تر را نشان می‌دهد. LC نیز مجموع دو مقدار قبلی را نشان داده که در آن LC بالا در تمام جفت گره‌های متوسط در DAG قابل مشاهده می‌باشد. با توجه به مقادیر LC، تمامی فعالیت‌ها و یا وظایف بصورت کاهشی توسط مقادیر LC قابل مرتب سازی خواهد بود. بنابراین فعالیت یا وظایفی که دارای مقادیر بیشتری از LC می‌باشند ابتدا اجرا گردیده و پیش از سایر فعالیت‌ها اجرا خواهند شد. در فاز انتخاب پردازنده، پردازنده‌ای که دارای کمترین زمان اولیه بوده انتخاب خواهد شد.

ILavarasan و همکارانش[۱۷] همچنین الگوریتم PETS را نیز مورد استفاده قرار داده که این الگوریتم فازی شبیه به فاز الگوریتم HPS دارد. همانند HPS تمام وظایف و فعالیت ها براساس سطح آنها طبقه‌بندی می-گردند. سپس، در فاز اولویت‌بندی، اولویت‌ها براساس ACC، PTC و RPT محاسبه خواهند شد. ACC شامل مجموع محاسبات تمام پردازنده‌های موجود می‌باشد. DTC مقادیر ارتباطی بین پردازنده و فعالیت مربوطه را نشان

می‌دهد. RPT بیشترین میزان جفت گره‌های حد واسط را نشان خواهد داد. بنابراین، مرتب سازی در فاز اولویت‌بندی با افزودن ACC، PTC و RPT امکان پذیر خواهد بود. سپس، فعالیت‌ها و یا وظایف مربوطه از بیشترین مقدار به کمترین مقدار مرتب‌سازی خواهد شد. همانند اکثر الگوریتم‌های پیشنهادی قبلی، در انتخاب پردازنده بایستی دقت نموده و نسبت به انتخاب پردازنده ای که حداقل EFT را داشته باشد اقدام خواهیم نمود.

Dai و همکارانش[۷] الگوریتم HCPPEFT را در نظر گرفته که از ۳ سطح اولویت بندی جهت انتخاب فعالیت استفاده می‌نمود. سطح اول برای فعالیت های اصلی بوده سطح دوم برای مسیرهایی که طول مسیر از گره خروجی تا فعالیت طولانی‌تر بوده و سطح سوم نیز برای فعالیت‌هایی که جایگزین کمتری دارند مورد استفاده قرار می‌گرفت. بنابراین، در انتخاب پردازنده جهت تکثیر فعالیت بایستی سعی بر این داشته تا طول زمان بندی به حداقل رسد.

براساس مطالعات اخیر که توسط shetti و همکارانش[۱۵] صورت گرفت نشانگر نوعی مکانیسم غیرمتقاطع بوده که باعث تقویت محیط GPU-CPU می‌گردد. محققان سعی بر این داشته تا با انتخاب بهینه پردازنده که با کاهش تقاطع بین CPU و پردازنده GPU صورت می‌گیرد، محیط CPU-GPU تقویت شود. علاوه بر این، محققان از نوعی تابع یا بردار مرتب‌سازی بهره گرفته که در این راستا از نسبت اجرایی آهسته ترین پردازنده تا سریع ترین پردازنده استفاده می‌گردد. با توجه به این اینکه الگوریتم‌ها دارای پیچیدگی یکسانی می‌باشند، و به عبارتی $O(t^2P)$ می باشد، الگوریتم پیشنهادی کاهش چشمگیری را در زمان اجرایی نشان می‌دهد. بنابراین، در این مطالعات اخیر، زمان ارتباطی بین فعالیت‌ها و یا وظایف جهت تعیین ترتیب سازی مورد استفاده قرار نمی‌گیرد.

همانند HEFT، الگوریتم پیشنهادی Imtiaz Ahmad و همکارش[۲] نیز دارای دو فاز بوده که به شرح ذیل است:

فاز اولویت بندی و فاز انتخاب پردازنده. بایستی توجه داشته باشیم که در فاز اولویت‌بندی مقادیر موجود در محاسبات معادلات برگشت‌پذیر با در نظر گرفتن مقادیر مطلوب سراسری و کلی (موضعی و سراسری) امکان پذیر

خواهد بود. این مکانیسم می تواند در فعالیت های کوچکتر اثرگذار بوده و اولویت بندی به ترتیب از بالا به پائین صورت گیرد. در فاز انتخاب پردازنده، به جای استفاده از مقادیر ثابت، از روشی که توسط shettie و همکاران مورد بررسی قرار گرفته استفاده شده است، که در این راستا از مرتب سازی و ترتیب تصادفی جهت جستجوی بهتر فضا استفاده کرده‌اند. در این تحقیق، از مکانیسم متقاطع جهت افزایش و تقویت الگوریتم پیشنهادی بهره گرفته شده است. از آنجا که از نتایج حاصل شده از این روش به عنوان جمعیت ورودی در الگوریتم پیشنهادی ارایه شده در این پژوهش استفاده شده است، توضیحات کامل‌تر در بخش سوم ارایه خواهد شد.

در الگوریتم CPOP[۱۶]، مجموع کل پارامترهای b_{level} و t_{level} هر یک از گره‌ها براساس اولویت‌بندی فعالیت در نظر گرفته شده و سپس فعالیت ها یا عملکردها براساس اولویت بندی انتخاب گردیده و اگر در مسیر اصلی قرار گرفته شده باشد با کمترین میزان محاسبات در اختیار پردازنده قرار گرفته شده و اگر در این مسیر قرار نداشته باشند، به گونه ای به پردازنده اختصاص داده می شوند که نزدیکترین زمان ممکن جهت اجرا را داشته و باعث ایجاد نزدیکترین زمان اجرایی خواهند شد. مسیر اصلی شامل مسیر ورودی و خروجی بوده که شامل بیشترین میزان محاسبات و لبه‌های ارتباطی می باشد. بنابراین، یک الگوریتم زمانبندی که براساس لیست بندی در نظر گرفته می‌شود مستلزم اولویت‌بندی فعالیت‌های موجود و زمانبندی در مسیر اصلی می باشد. طول مسیر اصلی برابر با مجموع پارامترهای b_{level} و t_{level} ورودی یا فعالیت مربوط به ورودی می باشد. بنابراین، هر یک از عملکردهایی که b_{level} و t_{level} آنها معادل مجموع b_{level} و t_{level} فعالیت ورودی باشد، در مسیر اصلی قرار خواهد گرفت.

در زمانبندی‌های جدید سعی شده تا با تخصیص پردازنده‌ای که گره پدر را پردازش کرده است به گره فرزند، از هزینه ارتباطات یا به عبارتی هزینه ناشی از انتقال نتایج گره والد، کاسته شود. این امر باعث می گردد تا از انتقال نتایج از یک فعالیت خاص و معین به فعالیت بعدی جلوگیری گردد (زیرا که هر دو فعالیت در یک پردازنده واحدی صورت می‌گیرد) و بنابراین، این امر باعث کاهش ارتباطات نیز خواهد شد. در سیستم توزیع و موازی، روش

نوین خوشه‌سازی نوعی روش و راه حل مناسبی بوده که از جهت کاهش هزینه ارتباطی می توان از آن استفاده نمود. تاخیر ارتباطی در این روش زمانی کاهش می‌یابد که در این روش هر یک از فعالیت ها ارتباط بسیار نزدیکی به یکدیگر داشته و تقریباً مشابه یکدیگر بوده و به یک پردازنده واحدی اختصاص داده می شود [۱]. در شکل۲-۳ می‌توان این گروه‌بندی را به همراه نمونه‌های تحقیق شده هر گروه مشاهده نمود.

الگوریتم‌های جستجوی تصادفی به‌طور معمول بهترین راه‌حل ممکن را جهت بهینه‌سازی عمومی مسائل طراحی می‌نماید. روش‌های جستجوی تصادفی نیز به عنوان روشی سطح بالاتر بوده و نسبت به روش مبتنی بر لیست پیشرفته‌تر بوده و اصلاحات یا تغییراتی روی آن انجام یافته است که بر این اساس می توان به زمانبندی بهینه نزدیکتر شد و با استفاده از روش جستجوی تصادفی به راه‌حل ممکن و قابل دسترسی دستیابی خواهیم داشت. در حالت کلی دو حالت مطلوب و بهینه وجود داشته که شامل حالت مطلوب سراسری و حالت مطلوب موضعی یا محلی می‌باشد که در بهینه‌سازی مسائل و موضوعات اثرگذار می‌باشد. مطلوب موضعی یا حالت مربوط به بهینه‌سازی محلی بهترین روش جهت مسائل و موضوعات مربوط به زیر مجموعه‌ها بوده ولی شامل تمامی مسائل مربوط به فضا نمی باشد. در برابر این امر، حالت مطلوب یا بهینه سازی سراسری در بر گیرنده تمام مسائل مربوط به فضا می‌باشد. پیدا کردن بهترین روش بهینه‌سازی سراسری برای تمام مسائل مربوط به فضا بسیار مشکل بوده و لذا بر این اساس عمدتاً همیشه سعی بر این گردیده تا بهترین روش مورد انتخاب قرار گرفته و استفاده شود. در این راستا از روش جستجوی تصادفی برای بهینه سازی استفاده گردیده و از الگوریتم آن بهره گرفته می‌شود. دلایل استفاده از این الگوریتم شامل ۳ گروه‌بندی بوده و به ۳ دلیل اصلی که عمدتاً سادگی، قابلیت انعطاف و قابلیت همزمانی در اجرا است، از الگوریتم جستجوی تصادفی استفاده می گردد[۱].

سادگی simplicity: اکثر الگوریتم‌های جستجوی تصادفی بسیار ساده بوده و کاربرد آسانی دارند و از پیچیدگی کمتری برخوردار می‌باشند. قسمت و بخش اولیه الگوریتم در ۱۰۰ خط به زبان برنامه نویسی نوشته می‌شود.

انعطاف پذیری Flexibility: الگوریتم ها بسیار انعطاف پذیر بوده و به قدری قابلیت انعطاف داشته که می‌توانند محدوده بسیار وسیعی از مسائل و موضوعاتی را که توسط الگوریتم‌های کلاسیک قابل حل شدن نبوده را تحت پوشش قرار دهند.

Ergodicity: این ویژگی الگوریتم‌های جستجوی تصادفی باعث می‌گردد تا چندین جستجوی همزمان را با موضوعات مختلف و متفاوت بدون کاهش راندمان و بازدهی در یک بازه زمانی و بصورت همزمان انجام دهند. این ویژگی بطور معمول حاصل روش تصادفی سیستم اصلی بوده و به عبارتی از مدل‌های آماری، جهشی و یا متقاطع برای مثال Random walk و یا Levy flight حاصل می گردد.

روش‌های مختلفی جهت حل مسائل مربوط به زمانبندی براساس متدهای جستجوی تصادفی وجود دارد که در قسمت زیر به مشهورترین و مطرح ترین روش ها اشاره شده است:

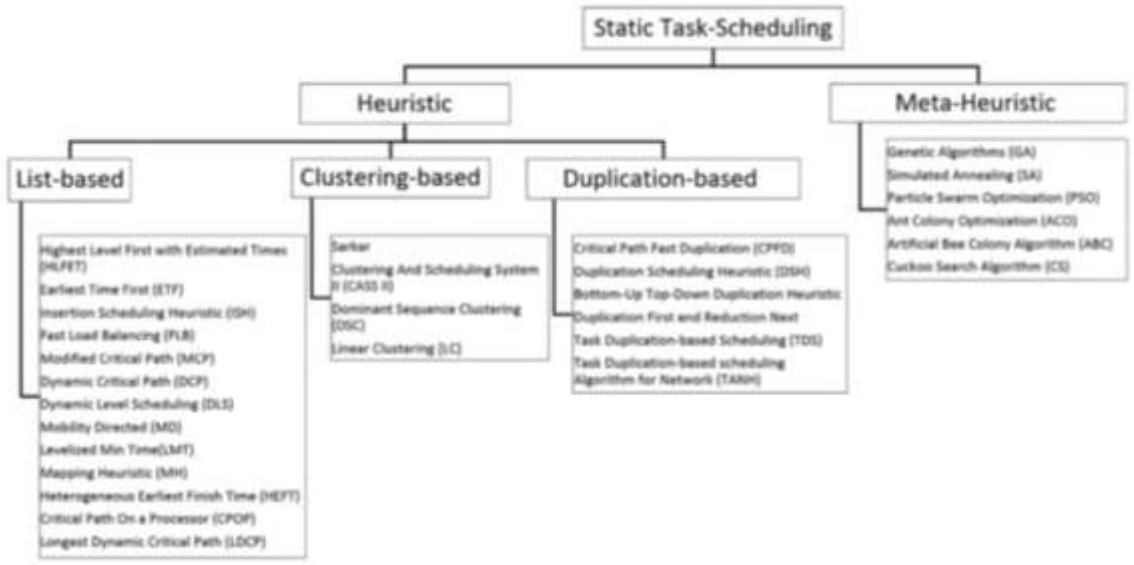
- الگوریتم هایی که براساس ژنتیک فعالیت می‌نمایند.
- بهینه سازی ذرات (PSO)
- بهینه سازی کلونی مورچه ای (موت) (ACO)
- الگوریتم جستجوی Cuckoo (CS)
- کلونی زنبور عسل مصنوعی (ABC)
- الگوریتم شبیه سازی حرارتی (Simulated Annealing)
- الگوریتم ممتیک (ممتازی)

در جدول ۳ فهرستی از الگوریتم‌ها و تعاریف مربوط به آنها که در این تحقیق مورد استفاده قرار گرفته ارائه شده است. پیچیدگی و بازه زمانی SA شامل $O(\tau XL \times Ln|R|)$ بوده که در آن τ نشانگر بازه زمانی و فاصله زمانی بوده و L بزرگترین حد آستانه (بزرگترین مجاورت موجود) و R نیز نشان دهنده ترکیب راه حل های موجود یا

ممکن خواهد بود. پیچیدگی و بازه زمانی PSO شامل O(M×K×LogK) بوده که در آن M نشان دهنده تعداد بردارهای تابع بوده و K نیز راه‌حل‌های موجود در آرشیو را نشان می دهد.

الگوریتم پیشنهادی تمایل شدیدی برای به حداقل رساندن زمان اجرا داشته و در راستای مطابقت هر یک از عملکردها با پردازنده جهت موازی سازی هر چه بیشتر و به حداکثر رساندن توازن حرکت نموده و از طرفی نیز باعث کاهش میزان ارتباطات جهت اجرای هر یک از فعالیت های موجود در پردازنده نیز خواهد شد. همچنین کاهش تعداد تکرار نیز در این الگوریتم حائز اهمیت بوده که این امر با کاهش تکرار یا حذف برنامه زمابندی‌های مشابه حاصل می‌گردد. جهت اطمینان از تنوع نمونه و مثال ها و پوشش ثابت و یکنواخت فضا، بایستی برخی تغییرات قابل ملاحظه‌ای بر روی تابع تولید همسایه صورت گرفته است.

در اکثر الگوریتم‌هایی که براساس لیست‌بندی فعالیت می نمایند، مرحله‌ای تحت عنوان مرحله انتخاب و مطابقت فعالیت با پردازنده وجود دارد. بنابراین، روش جدید در الگوریتم پیشنهادی مورد استفاده قرار گرفته شده تا بر این اساس زمانبندی هر یک از فعالیت‌های مرتبط با پردازنده مشاهده گردیده و تعیین شود و زمانبندی الگوریتم در مرحله ایجاد جمعیت اولیه معین شود.



شکل ۲-۳- برخی الگوریتم‌های استفاده شده در حل مساله

جدول۲- ۳- زمان اجرای الگوریتم‌ها

Algorithm	Type	Priority	Time complexity
HEFT-B	Static list scheduling +Insertion-based	Blevel	$O(n^2 \times p)$
HEFT-T	Static list scheduling +Insertion-based	Tlevel	$O(n^2 \times p)$
CPOP	Static list scheduling	blevel+tlevel	$O(n^2 \times p)$
BGA	Genetic	blevel+tlevel	$O(\text{generations} \times n^2 \times p)$
SA	Simulated annealing	Modified Best Fit (MBF)	$O(\tau \times L \times \ln \lvert R \rvert)$
SLPSO	Particle Swarm Optimization	The personal best (pbest)	$O(M \times K \times \log K)$
EGA-ST	Genetic	blevel+tlevel	$O(\text{generations} \times n^2 \times p)$

فصل ۳: روش پیشنهادی

دراین بخش ما با ترکیب دو تکنیک حل مساله زمانبندی وظیفه در سیستم‌های ناهمگن، روشی نوین برای این کار پیشنهاد داده‌ایم. مانند الگوریتم‌های پیشین هدف اصلی کمینه کردن زمان اتمام تمامی کارها با در نظر گرفتن هزینه جا به جایی وظایف در بین پردازنده‌ها و وابستگی میان وظایف می‌باشد. الگوریتم پیشنهادی از دو بخش پیدا کردن راه حل نزدیک به بهینه و بهبود راه حل تشکیل شده است. در این بخش، نخست الگوریتم توسعه داده شده توسط Ahmad and Al Ebrahim توضیح داده شده و با الگوریتم HEFT مقایسه می‌گردد. این الگوریتم پایه الگوریتم پیشنهادی و خروجی آن به عنوان جمعیت اولیه در بخش دوم استفاده خواهد شد. در مرحله دوم نتایج به دست آمده در مرحله اول با استفاده از الگوریتم توسعه داده شده توسط تکنیک شبیه‌سازی تبرید در چندین تکرار بهبود داده خواهد شد. در آخر نیز نتایج به دست آمده در ۳ حالت زمانبندی فقط با الگوریتم Ahmad and Al Ebrahim، زمانبندی با استفاده از تکنیک شبیه‌سازی تبرید با ورودی تصادفی و زمانبندی با استفاده از تکنیک شبیه‌سازی تبرید با ورودی جمعیت زمانبندی شده توسط الگوریتم Ahmad and Al Ebrahim مقایسه خواهند شد.

۳-۱ الگوریتم ورودی اولیه

الگوریتم پیشنهادی در بخش ورودی اولیه به دو قسمت اولویت‌بندی کار و انتخاب پردازنده تقسیم می‌شود. در فاز اولویت‌بندی، وظایف با توجه به اولویتشان رده‌بندی می‌شوند. سپس با توجه به اولویت داده شده به هر وظیفه پردازنده مناسب برای اجرا انتخاب می‌شود. وظایف با توجه به وزن محاسبه شده از زمان اجرا، زمان یا هزینه ارتباط و وزن وظایف قبلی محاسبه می‌شود.

۳-۱-۱ فاز اولویت‌بندی

الگوریتم کار خود را با محاسبه $weight_{ni}$ برای محاسبه اولویت وظایف شروع می‌کند. $weight_{ni}$ برای تمامی وظایف درون گراف حساب می‌گردد. $Weight_{ni}$ مقدار مطلق ضریب اختلاف بین پردازنده با بیشترین هزینه زمانی و کمترین هزینه زمانی است:

$$Weight_{ni} = | (w(ni,pj) - w(ni,pk)) / ((w(ni,pj) / w(ni,pk)) |$$

(۱-۳)

که در آن w(ni,pk) زمان اجرای وظیفه i در پردازنده j می‌باشد. انتخاب‌های دیگری برای $weight_{ni}$ مانند میانگین، میانه، بیشینه و کمینه نیز وجود دارد ولی فرمول ارایه شده دارای اجراهای بهتری است.

سپس برای تکمیل فاز اولویت بندی و رده‌بندیوظایف به صورت پایین به بالا گراف را پیموده و با توجه به اولویت‌ها و محدودیت‌ها و $weight_{ni}$ وظایف را طبق فرمول رده بندی می‌کنیم:

$$Rank_{proposed(ni)} = weight_{ni} + MAX_{nj \in succ(ni)}\{AVG(C_{i,j}) + Rank_{proposed(nj)}\}$$

(۲-۳)

۳-۱-۲ – انتخاب پردازنده

در این فاز یک تغییر عمده نسبت به HEFT وجود دارد آن‌هم استفاده از یک عملگر تصادفی برای تعویض پردازنده می‌باشد. برخلاف HEFT و PEFT این تکنیک باعث می‌شود تا پردازنده با زمان اتمام کار کمتر انتخاب نشود بلکه وظیفه با استفاده از نتیجه محاسباتی که آن را آستانه می‌نامیم با ضریب احتمال بین صفر و یک به پردازنده دیگر که زمان پردازش کوتاه‌تر دارد محول شود[۲]. هرچه آستانه به یک نزدیکتر باشد رفتار الگوریتم به HEFT شبیه‌تر خواهد بود. در طرف دیگر هرچه به صفر میل کند امکان تعویض پردازنده کمتر خواهد شد. به عبارت دیگر عمل تعویض پردازنده یعنی دادن وظیفه به پردازنده با زمان پردازش کمتر به جای پردازنده با زمان اتمام پردازش نزدیکتر. این کار باعث کمتر شدن زمان اتمام تمامی وظایف می‌شود. الگوریتم با این کار، با تجمیع تصمیمات بهینه محلی و بهینه سراسری فضای جستجو را برای پیدا کردن زمانبندی کمینه بهتر و موثرتر جستجو می‌کند.

توجه داشته باشید که عمل تعویض پردازنده در حین اجرای وظیفه صورت نمی‌پذیرد. به صورت کلی در این مقاله از زمانبندی‌هایی سخن گفته شده و می‌شود که عمل تخصیص منابع به صورت با قبضه‌ای بوده و وظیفه نمی‌تواند حین اجرا متوقف و یا به دیگر پردازنده‌ها منتقل شود.

در الگوریتم ۳-۱ به‌صورت فرمال پروسه زمانبندی کار که در Ahmad and Al Ebrahim آمده و پایه و جمعیت الگوریتم پیشنهادی است، نشان داده شده است. در خطوط ۱ تا ۳ الگوریتم، نخست رتبه (ni) با توجه به محاسبات $weight_{ni}$ حساب شده است. سپس وظایف در لیست آماده ذخیره می‌شوند. لیست آماده یک صف به صورت زودترین ورودی، زودترین خروجی محاسبه شده با اولویت‌هایی که قبل‌تر گفته شده است. در مرحله بعد تا زمانی که لیست آماده خالی شود، وظایف را به پردازنده‌ها اختصاص می‌دهیم. بنابراین، با استفاده از یک ترتیب کاهشی، در خطوط ۴ تا ۸ نزدیکترین زمان شروع پردازش برای هر پردازنده محاسبه می‌شود. در محاسبه نزدیکترین زمان اجرا باید به این مساله توجه داشت که زمان انتقال داده در وظایفی که پردازش والد در پردازنده دیگری صورت گرفته باید به زمان شروع پردازش زمان انتقال نیز افزوده شود.

پس از آن پردازنده با زمان اتمام کار انتخاب شده و با پردازنده‌ای که دارای زمان اجرای کمتری است مقایسه می‌شود. اگر پردازنده انتخاب شده پردازنده با کمترین زمان اجرا باشد یا به عبارت دیگر شرط خط ۱۰ برقرار باشد، بنابراین کمینه محلی با کمینه سراسری در این پردازش برابر است و پردازنده انتخاب شده به وظیفه اختصاص داده می‌شود و لیست آماده با زمان اجرا و زمان شروع به پردازش به روز می‌شود. اگر پردازنده انتخاب شده کمترین زمان اجرا برای این وظیفه را در بین بقیه پردازنده‌ها نداشته باشد و شرط خط ۱۰ برقرار نباشد، برای انتخاب پردازنده باید وزن مطلق و آستانه محاسبه گردد. وزن مطلق مقدار زمان ذخیره شده از افزایش سرعت پردازش می‌باشد.

$$Weight_{abstract} = | (EFT(ni,pj) - EFT(ni,pk)) / (EFT(ni,pj)/EFT(ni,pk)) |$$

(۳-۳)

```
begin
1.    for all n_i in N do
2.          Compute rank_proposed(n_i);
3.    end
4.    while ready-list is NOT Empty do
5.          n_i ← task in the ready-list with the maximum rank;
6.          for all p_j in P do
7.                Compute EFT(n_i,p_j) ← w_i,j + EST(n_i,p_j);
8.          end
9.          Select p_j with Min EFT(n_i,p_j);
10.         if w_i,j ≤ Min k ∈ P(w_i,k) then
11.               Assign n_i to the processor p_j that minimize EFT of task n_i;
12.               Update T_available(p_j) and ready-list;
13.         else
14.               Compute Weight_abstract and Cross_Threshold;
15.               if Cross_Threshold ≤ r then          // r is a random number
16.                     Map node n_i to processor p_j;      //cross over
17.                     Update T_available(p_j) and ready-list;
18.               else
19.                     Map node n_i to processor p_k;      //no cross over
20.                     Update T_available(p_k) and ready-list;
21.               end
22.         end
23.   end
```

الگوریتم ۳-۱ – الگوریتم پیشنهادی Ahmad and Al Ebrahim

سپس با استفاده از فرمول ذیل آستانه محاسبه گردد:

$$CROSS\_THRESHOLD = weight_{ni} / weight_{abstract}$$

(۴-۳)

اگر مقدار آستانه‌ای کمتر یا مساوی مقدار تصادفی در رنج ۰,۱ تا ۰,۳ باشد، وظیفه به پردازنده با کمترین زمان اجرا داده خواهد شد . این کار باعث یک تصمیم تصادفی برای جستجوی موثرتر در فضای راه‌حل‌ها خواهد شد. از طرفی اگر مقدار آستانه از عدد تصادفی بیشتر باشد تغییر پردازنده صورت نخواهد گرفت و وظیفه در همان

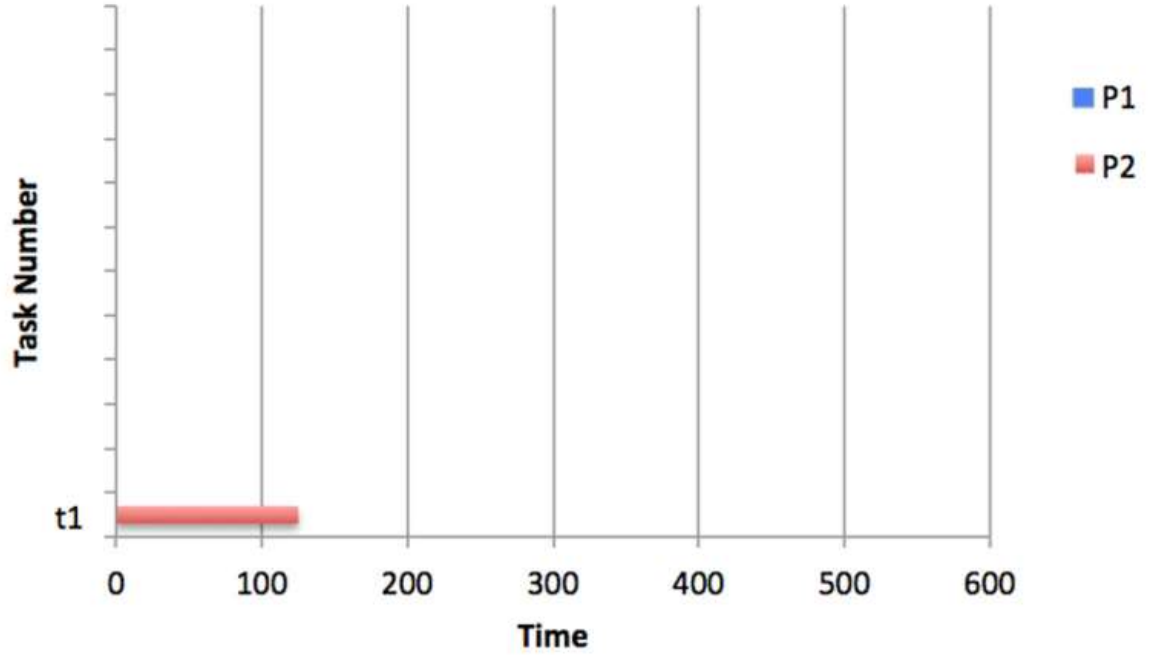
پردازنده با زمان اتمام کار کمتر پردازش خواهد شد. به مانند الگوریتم‌های HEFT و PEFT در این الگوریتم نیز زمان $O(t^2p)$ صادق است. T تعداد وظایف و p تعداد پردازنده‌ها می‌باشد.

۳-۱-۳ – مثال تشریحی

برای آزمایش الگوریتم روی مثال تصویر ۱، نخست شروع به محاسبه $rank_{proposed(ni)}$ برای هر وظیفه می‌کنیم. از t10 شروع و تا t1 ادامه می‌دهیم. توضیح مقادیر در جدول نمایش داده شده است. خروجی مکانیزم رده‌بندی به صورت صف اولویتی {t1,t5,t4,t2,t6,t8,t3,t9,t7,t10} است. با مقایسه رده‌بندی این الگوریتم با الگوریتم‌های HEFT و PEFT که قبلا گفته شد، متوجه می‌شوید وظیفه شماره ۸ اولویت بالاتری گرفته است. تغییر اولویت به این دلیل است که برخلاف دو الگوریتم دیگر، در این الگوریتم محاسبات فقط بر حسب میانگین زمان اجرا صورت نمی‌پذیرد و این کار باعث می‌شود وظایف دارای زمان اجرای بالاتر هستند به تاخیر انداخته نشوند و شانس بیشتری داشته باشند.

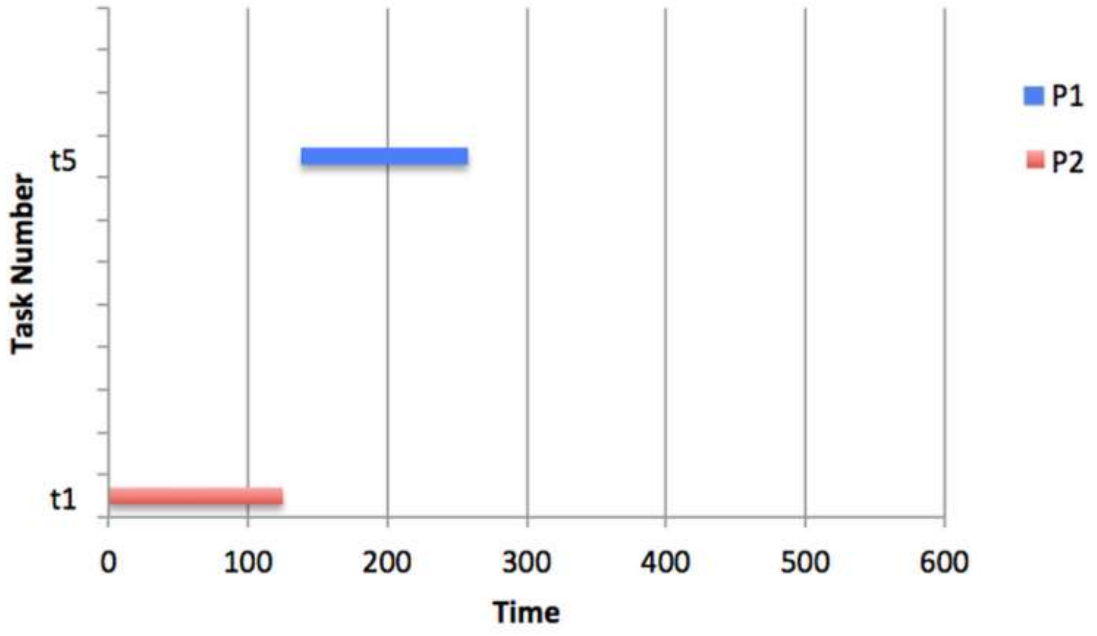
جدول ۳-۱ اولویت بندی در Ahmad and Al Ebrahim

Tasks	rankproposedt (ni)	Processor selected	Earliest start time (EST)	Earliest finisl time (EFT)
t1	195.5	2	0	125
t5	149.0	1	138	258
t4	119.5	1	258	403
t2	97.0	2	125	239
t6	87.0	1	403	413
t8	73.0	1	413	463
t3	68.5	2	238	370
t9	50.5	2	410	475
t7	32	2	475	505
t10	0.5	2	505	507



شکل ۳-۱ اجرای t1 در پردازنده ۲ تحت الگوریتم Ahmad and Al Ebrahim

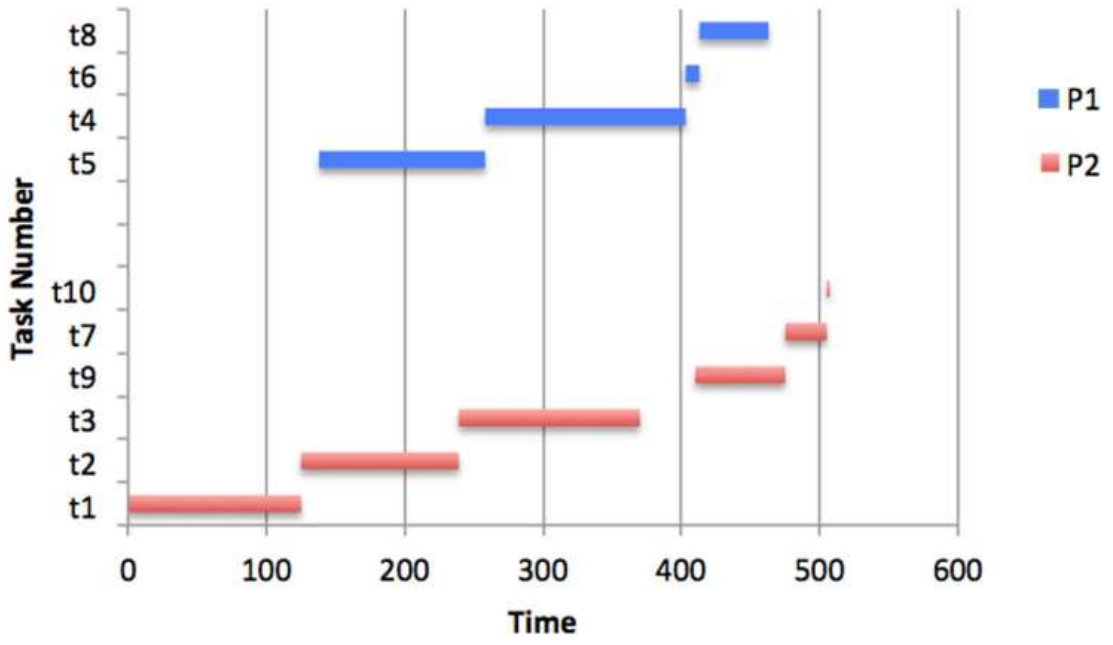
در فاز بعدی، در شروع هر دو پردازنده در دسترس هستند. بنابراین زمان شروع برای t1 در هر دو پردازنده برابر با صفر است. EFT(t1,p1) برابر با ۱۷۱ و EFT(t1,p2) برابر با ۱۲۵ است. به علت پردازش سریعتر پردازنده دوم، این پردازنده انتخاب می‌شود.همانطور که در تصویر ۴ مشاهده می‌شود پردازش t1 در p2 از صفر تا واحد زمانی ۱۲۵ طول می‌کشد. چون t1 والد تمامی وظایف بعدی می‌باشد و بقیه وظایف به این وظیفه وابسته‌اند، بنابراین حداقل زمان شروع برای بقیه وظایف واحد زمانی ۱۲۵ خواهد بود.



شکل ۳-۲ اجرای t2 در پردازنده ۱ تحت الگوریتم Ahmad and Al Ebrahim

اولویت بعدی برای پردازش t5 است. EST(t5,p1) برابر ۱۳۸ و برای EST(t5,p2) برابر ۱۲۵ است. بنابراین EFT(t5,p1) مساویست با ۲۵۸ و EFT(t5,p2) مساوی ۳۰۹ خواهد بود. از آن جا که p1 دارای زمان اجرای کوتاهتر و p2 دارای زمان اتمام کمتر است، p1 که دارای زمان اجرای کمتر است انتخاب می‌شود.

طبق لیست نوبت به t4 می‌رسد. EST(t4,p1) مساوی ۲۵۸ و EST(t4,p2) برابر ۱۲۵ است. پس EFT(t4,p1) برابر ۴۰۳ و EFT(t4,p2) برابر ۳۱۷ خواهد بود. از آنجا که p1 دارای زمان اجرای کوتاهتر و p2 دارای زمان اتمام کمتر است، باید محاسبات آستانه برای تعیین پردازنده صورت گیرد. اگر عدد تصادفی را ۰,۳ و نتیجه محاسبات آستانه که برابر ۰,۵ است را در نظر بگیریم، بنابراین تعویض پردازنده صورت نمی‌گیرد و t4 در p1 اجرا خواهد شد.



شکل ۳-۳ زمانبندی تمام وظایف تحت الگوریتم Ahmad and Al Ebrahim

الگوریتم به همین منوال برای t2، t6 و t8 ادامه خواهد داشت. نوبت به t3 خواهد رسید. برای این وظیفه p1 پردازنده با کمترین زمان اجرا و p2 پردازنده با کمترین زمان اتمام کار است. برای این وظیفه نیز آستانه محاسبه خواهد شد. آستانه محاسبه شده برابر ۰,۲۳ است و عدد تصادفی را مانند قبل ۰,۳ در نظر می‌گیریم. از آن‌جایی که عدد تصادفی مقدار بیشتری دارد، لذا پردازنده p2 به جای p1 انتخاب خواهد شد.

وظایف t9، t7، t10 نیز بر روی پردازنده ۲ اجرا خواهند شد. با توجه به مکانیزم محدود شده انتقال بین پردازنده‌ها، زمان پایان کل وظایف ۵۰۷ واحد زمانی خواهد بود. بنابراین وظایف {t1,t2,t3,t9,t7,t10} در پردازنده دوم و وظایف {t5,t4,t6,t8} در پردازنده یک پردازش خواهند شد.

به طور خلاصه، مکانیزم تعویض پردازنده نتایج بهتری علی‌الخصوص در برنامه‌هایی با وظایف طولانی‌تر حاصل می‌کند. در مقایسه HEFT، PEFT و این الگوریتم به ترتیب ۵۲۱، ۵۳۷ و ۵۰۷ به دست آمده است.

۲-۳ – بهبود نتایج با استفاده از تکنیک شبیه‌سازی تبرید

روش جستجوی تبرید شبیه سازی شده (SA) یک جستجوگر همسایگی است که در بهینه‌سازی مسائل گسسته به طور گسترده‌ای کاربرد دارد. طبیعت تصمیم‌گیری این الگوریتم به این صورت است که برای هر حرکت، یک همسایگی جدید به صورت تصادفی تولید و ارزیابی می‌شود .

الگوریتم شبیه سازی تبرید (SA) روشی ابتکاری بر مبنای جستجوی محلی است. این الگوریتم قادر است از گیر افتادن در بهینه محلی با قبول جواب های بدتر، با احتمالی کم، جلوگیری کند. کیفیت جواب های حاصل از الگوریتم شبیه سازی تبرید موجب شده تا در حل مسائل بهینه سازی ترکیبی پیچیده در حوزه های مختلف مسائل دنیای واقعی مورد استفاده قرار گیرد. شبیه سازی تبرید توسط کرک پاتریک و همکاران عمومیت پیدا کرد. مفهوم اصلی شبیه سازی تبرید از فرایند تبرید فلزات در صنعت متالورژی گرفته شده است .

فرایند بهینه سازی در شبیه سازی تبرید، جستجو برای یک جواب (نزدیک به) کمینه سراسری است. الگوریتم از یک جواب تصادفی به عنوان جواب اولیه حرکت خود را آغاز میکند و دمای سیستم برابر دمای اولیه قرار میگیرد (T=T) .در هر تکرار یک جواب همسایه جواب فعلی به دست می آید. مقدار تابع هدف جواب جدید با جواب فعلی مقایسه می شود. اگر جواب بهتر بود، جایگزین جواب فعلی می شود و اگر جواب بدتر بود با یک احتمال که از تابع بالتزمن $\exp(-\Delta/KT)$ به دست می آید، جایگزین جواب فعلی می شود. این مکانیزم منجر به این میشود که الگوریتم در بهینه محلی گیر نیافتد. در این رابطه برابر میزان اختلاف مقدار تابع هدف جواب فعلی با جواب جدید است، K ضریب بالتزمن است که از قبل تعیین می‌شود و T نیز دمای فعلی است. فرایند جستجوی همسایگی ادامه می یابد تا زمانی که تعداد تکرارها به مقداری از پیش تعیین شده برسد. پس از اینکه تعداد تکرارها به اندازه مقداری معین و از پیش تعیین شده گشت، دمای سیستم کاهش می یابد. این فرایند تا زمانی ادامه پیدا میکند که الگوریتم به معیار خاتمه برسد.

حرکت به این جواب در هر یک از دو وضعیت زیر انجام خواهد یافت :

۱. جواب جدید از جواب فعلی بهتر باشد

۲. .مقدار تابع احتمال حرکت [i] از یک عدد تصادفی از دامنه [۰،۱] بزرگتر باشد .

در غیر این صورت جستجوگر جواب جدیدی را تولید و ارزیابی خواهد نمود. این حرکت گام به‌گام تا ارضاء شرط توقف الگوریتم (تعداد تکرارها، زمان محاسبات، و ...) ادامه می‌یابد .

مقدار تابع احتمال حرکت در هر بار از رابطه محاسبه می‌شود. در این رابطه اختلاف مقدار تابع هدف بین جواب فعلی و جواب جدید است. شبیه سازی تبرید در واقع یک روش جستجوی تصادفی قوی است که به منظور یافتن یک جواب خوب (نه لزوما بهینه) برای مسائل مشکل ترکیباتی combinatorial به کار می رود. این روش برخلاف روش های جستجوی معمولی، در هر تکرار علاوه بر حرکت به سوی جواب بهتر، جواب های با مقدار تابع هدف بهتر را نیز با احتمال غیر صفری قبول می کند .شبیه سازی تبرید از فرایند فیزیکی خنک سازی مواد مذاب به حالت جامد الهام گرفته است.

در این مرحله از کار زمانبندی محاسبه شده در مرحله قبل به الگوریتم پیشنهادی داده می‌شود. برای این که جمعیت بیشتر باشد و امکان جستجوی بیشتری فراهم آید، از دو روش دیگر نیز که دارای پیچیدگی خطی O(t) هستند استفاده می‌شود. از بین زمانبندی‌ها، زمانبندی‌ای که دارای کمترین زمان اتمام تمامی کارها است را به عنوان بهترین راه‌حل و زمان‌بندی دوم را به عنوان راه‌حل اصلی در نظر می‌گیریم.

پس از تعیین بهترین راه‌حل الگوریتم به جستجو در همسایگان زمانبندی ها در فضای جستجو می‌کند. عمل جستجو دارای دو مرحله ترکیب و تغییر پردازنده است. در مرحله ترکیب به مانند عمل ترکیب در الگوریتم ژنتیک که روی ژن‌ها انجام می‌گرفت، این بار دو زمانبندی را با یکدیگر ترکیب می‌کنیم. برای این کار یک وظیفه

به صورت تصادفی انتخاب شده و فرزند آن در پردازنده مقیم با فرزندش در پردازنده دیگر جا به جا می‌شوند. باید توجه داشت عمل ترکیب با احتمال تصادفی بین ۰ تا یک صورت می‌گیرد برای آن آستانه ترکیب در نظر باید گرفت. بدیهی است هرچه آستانه به یک نزدیکتر باشد احتمال تولید همسایه بیشتر می‌شود.

عمل تغییر پردازنده نیز مانند ترکیب دارای آستانه احتمال بوده و برخلاف ترکیب که بر روی دو زمانبندی صورت می‌گرفت، بر روی یک وظیفه در یک زمانبندی صورت می‌گیرد. هدف از این کار کاستن از زمان انتقال داده در زمانبندی و در نهایت کاهش زمان اتمام کار از این طریق است. برای این کار در هر زمانبندی، با احتمال تصادفی بین ۰ و ۱ و آستانه تعیین شده از قبل، یک وظیفه که دارای بیشترین زمان انتظار است انتخاب شده و در پردازنده دیگر جایگذاری می‌شود.

بعد از ترکیب راه‌حل‌ها و اعمال تغییر پردازنده با استفاده از یک تابع ارزیابی، تمامی جمعیت را ارزیابی و نتایج ضعیف از لیست حذف می‌شود. در بین جمعیت باقیمانده بهترین راه‌حل به عنوان بهترین همسایه انتخاب می‌شود. حال با استفاده از اختلاف راه‌حل اصلی و بهترین همسایه، دلتا و از فرمول $(r - e^{\Delta T})$ احتمال انتخاب را حساب می‌کنیم. اگر هر دو مقذار بیش از صفر بود، بهترین همسایه به عنوان راه‌اصلی انتخاب می‌شود. در صورت انتخاب بهترین همسایه به عنوان راه‌حل اصلی، و در صورتی که بهترین همسایه راه‌حل بهتری از بهترین راه‌حل انتخاب شده باشد، بنابراین بهترین راه حل همسایه به عنوان بهترین راه‌حل در نظر گرفته می‌شود.

عمل تولید همسایه و انتخاب بهترین راه‌حل تا زمانی که شرط اتمام حلقه به ارضا رسیده و از حلقه خارج شود ادامه خواهد داشت و در هر تکرار دما کاهش خواهد داشت. پس از اتمام حلقه بهترین راه‌حل انتخاب شده به عنوان راه‌حل پایانی انتخاب خواهد شد.

فصل ۴: نتایج تحقیق

در این بخش گزارش مقایسه‌ای بین کارایی زمانبندی‌های HEFT، Al Ebrahim and Ahmad و الگوریتم پیشنهادی در این پژوهش با استفاده از دو متریک زمان اتمام تمام کارها (makespan) و بهبود طول زمان اجرایی زمانبندی پرداخته شده. نخست به تعریف متریک‌ها پرداخته می‌شود. سپس، محیط شبیه‌سازی شامل الگوریتم تولید گراف جهت دار بدون دور گفته شده و در آخر به آنالیز کارایی هر الگوریتم با استفاده از دو متریک گفته شده است.

۴-۱ – متریک های اندازه گیری

الگوریتم‌ها توسط دو متریک مقایسه خواهند شد:

الف) زمان اتمام تمام کارها:

یکی از متریک‌های پر استفاده در زمانبندی یک گراف جهت‌دار بدون دور، مقایسه زمان اتمام تمام کارها یا همان makespan است. زمان اتمام کارها یعنی زمانی که تمامی کارها انجام شده و تمام شده‌اند ودیگر وظیفه‌ای برای انجام موجود نیست.

$$Makespan = maxAFT(n_{exit})$$

(۱-۴)

ب) بهبود زمان طول اجرای زمانبندی

الگوریتم پیشنهادی به صورت میزان بهبود طول اجرا نسبت به دیگر الگوریتم ها نیز بررسی خواهد شد. ضریب بهبود طول اجرای برنامه از فرمول ذیل محاسبه می‌شود.

$$\% SL_{improvement} = (1 - (SL_{pr} / SL_A)) * 100$$

(۲-۴)

که در آن SL_{pr} نمایانگر طول اجرای زمانبندی با الگوریتم پیشنهادی و SL_a بیانگر طول اجرای زمانبندی در دیگر الگوریتم است. مقدار بالاتر ضریب SL نشانگر این است که الگوریتم پیشنهادی دارای نتایج بهتری بوده است.

۴-۲ – محیط شبیه سازی

علاوه بر الگوریتم پیشنهادی، HEFT و الگوریتم پیشنهادی Ahmad and Al Ebrahim در محیط php ورژن ۷,۲ در سیستم Lenovo ideapad z510 با سیستم‌عامل Ubuntu 16.04، پردازنده core i5 2.50 GHz و ۶ گیگابایت حافظه رم توسعه داده شده. تعداد بالای گراف نیز توسط سیستم تولید گراف تصادفی برای تست این الگوریتم در این محیط توسعه داده شده است. تمامی وزن و هزینه های ارتباطی به صورت تصادفی و با کمینه ۵ و بیشینه ۵۰ واحد زمانی برای آزمایش استفاده شده است.

۴-۳ – آنالیزها و نتایج کارایی هر الگوریتم

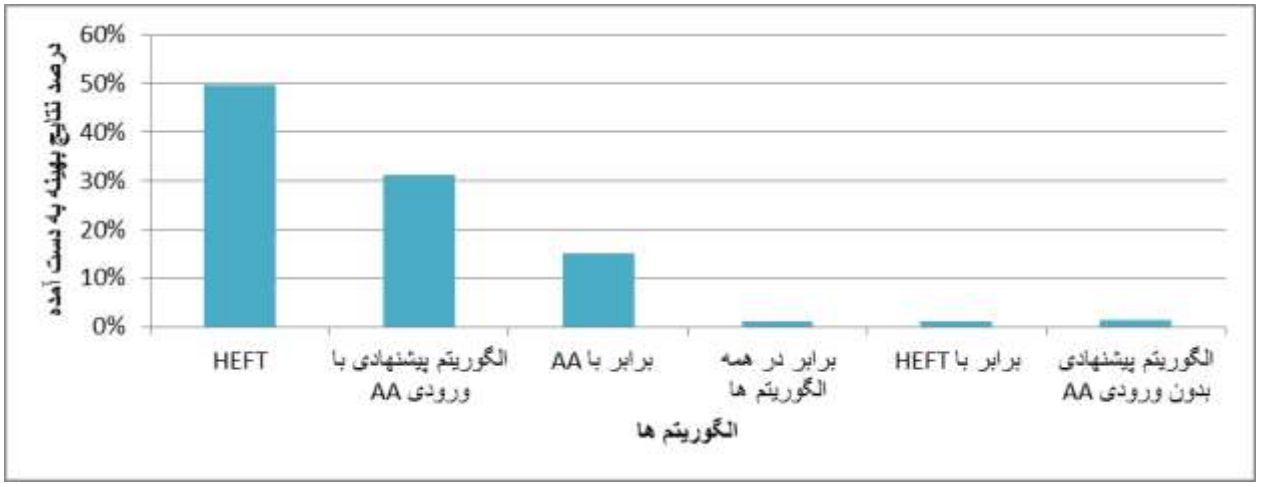
در این بخش به آنالیز نتایج الگوریتم‌ها پرداخته می‌شود که بر روی گراف‌های متنوع با تعداد گره مختلف آزمایش شده است. در هر آزمایش الگوریتم‌ها بر روی ۷۵۰ گراف اجرا می‌شوند و در نهایت الگوریتم‌ها براساس تعداد بهترین خروجی و میزان بهبود با یکدیگر مقایسه می‌شوند.

الف) makespan

کاهش زمان اتمام تمام کارها هدف اصلی تمام الگوریتم‌های زمانبندی‌ها می‌باشد. در آزمایشات مشاهده شد، هر چه تعداد وظایف بیشتر شود الگوریتم پیشنهادی بهبود بیشتری نسبت به دیگر الگوریتم‌ها خواهد داشت.

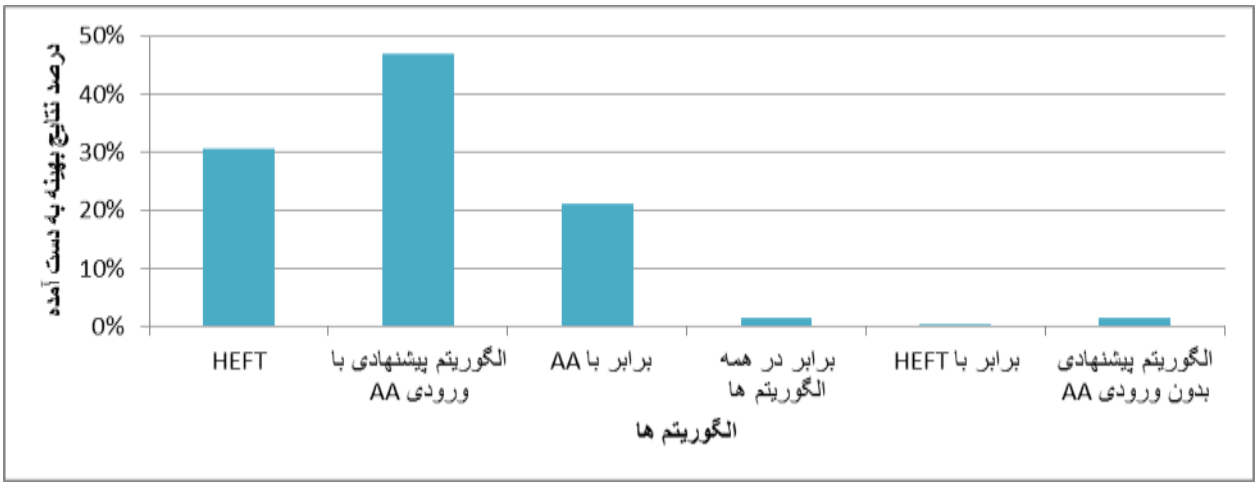
در شکل ۴-۱ نمودار اجرای الگوریتم‌ها بر روی گراف‌های ۵۰ گرهی آورده شده است. همانطور که مشاهده می‌شود در این تعداد از گره‌ها الگوریتم HEFT بیش از سایرین زمانبندی‌های با زمان اتمام کار کمتر را ایجاد کرده است. الگوریتم HEFT ۳۷۲ زمانبندی کمینه یعنی ۴۹,۶٪ کل آزمایشات به تنهایی بهترین جواب را یافته است. در مقابل الگوریتم پیشنهادی با ۳۴۹ زمانبندی کمینه که ۴۶,۳۳٪ کل آزمایشات است، بهتر عمل کرده است. در ۲۳۵ مورد از ۳۴۹ مورد مربوط به الگوریتم پیشنهادی، الگوریتم پیشنهادی بهتر از الگوریتم Ahmad and Al Ebrahim بوده و در بقیه ۱۱۴ مورد زمان یکسانی را ارایه کرده‌اند. از ۲۹ مورد باقی مانده در ۹ مورد تمامی زمانبندی‌ها زمان اتمام کار یکسانی داشته‌اند. در ۹ مورد از زمانبندی‌ها الگوریتم پیشنهادی بهتر از الگوریتم

Ahmad and Al Ebrahim و برابر با HEFT بوده است. در ۱۱ مورد باقی‌مانده الگوریتم پیشنهادی بدون استفاده از زمانبندی حاصل شده از الگوریتم Ahmad and Al Ebrahim زمانبندی کمینه را داشته است.



شکل ۴-۱– درصد تولید بهترین زمانبندی با ۵۰ وظیفه در الگوریتم‌های مختلف

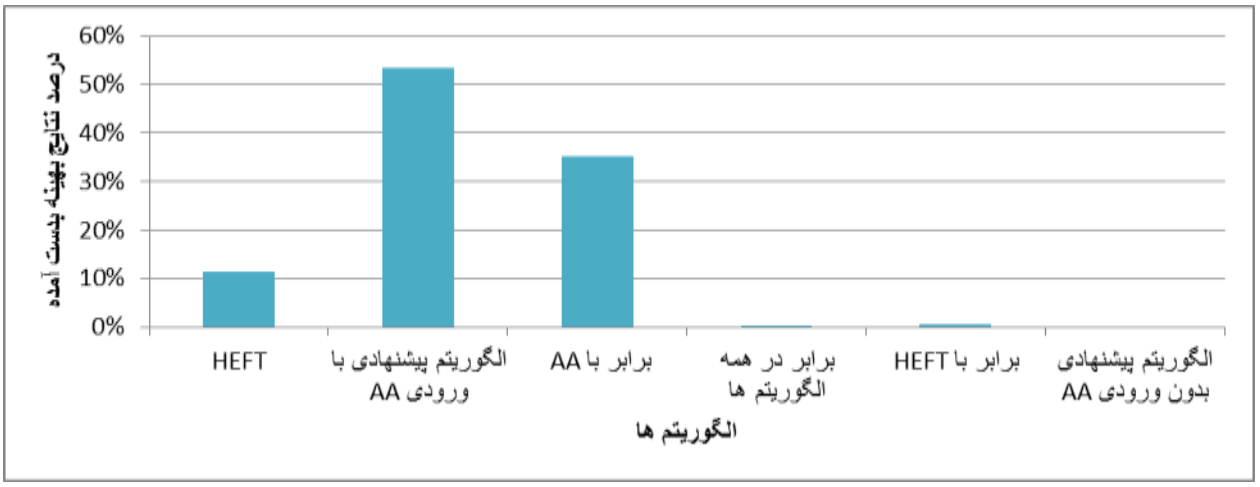
در نمودار ۴-۲ نمودار اجرای الگوریتم با در گراف‌های با ۱۰۰ گره آورده شده است. در این نمودار برخلاف نمودار آزمایش با ۵۰ گره مشاهده می‌شود الگوریتم پیشنهادی زمانبندی های کمینه بیشتری را تولید کرده است. در ۳۵۲ مورد (۴۶,۹۳٪) از آزمایشات الگوریتم پیشنهادی زمانبندی بهینه‌تری نسبت به باقی الگوریتم‌ها ایجاد کرده است. در ۲۲۹ (۳۰,۵۳٪) مورد از کل آزمایشات نیز الگوریتم HEFT پاسخ بهینه‌تر را ارایه می‌کند. در ۱۶۴ مورد از آزمایشات که ۲۱,۹٪ کل آزمایشات را در بر می‌گیرد نتیجه بهینه همان نتیجه‌ای است که الگوریتم Ahmad and Al Ebrahim به عنوان ورودی به الگوریتم شبیه سازی تبرید داده شده است. در ۳ مورد زمانبندی بهینه توسط الگوریتم پیشنهادی و HEFT مشترک بوده و در یک مورد توسط همه الگورینم‌ها زمانبندی یکسان به دست آمده است. فقط در یک مورد زمانبندی بهینه بدون استفاده از ورودی الگوریتم Ahmad and Al Ebrahim در الگوریتم پیشنهادی به دست آمده.



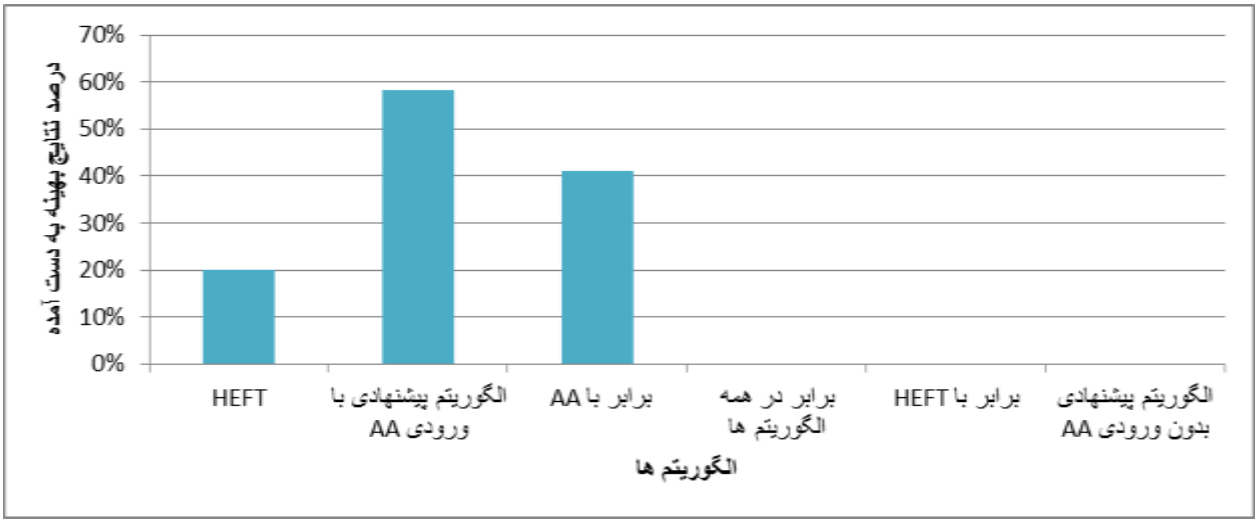
شکل ۴-۲ – درصد تولید بهترین زمانبندی با ۱۰۰ وظیفه در الگوریتم‌های مختلف

با بیشتر کردن تعداد گره وضعیت برای الگوریتم پیشنهادی در برابر HEFT بهتر می‌شود. در آزمایش الگوریتم‌ها در گراف‌های با ۲۰۰ گره، الگوریتم پیشنهادی ۴۰۱ بار راه‌حل بهینه را نسبت به HEFT و Ahmad and Al Ebrahim را ایجاد کرده است. این تعداد ۵۳,۴۷٪ کل آزمایشات است. ۳۵,۲۶٪ از کل آزمایشات که شامل ۲۵۸ آزمایش می‌شود طول زمانبندی الگوریتم پیشنهادی با طول زمانبندی الگوریتم Ahmad and Al Ebrahim برابر است. در ۸۵ مورد که ۱۱,۳۳٪ کل آزمایشات است HEFT راه‌حل بهتر را نسبت به بقیه زمانبندها ایجاد کرده است. در ۲ مورد نتایج تمام الگوریتم‌ها برابر و در ۴ مورد طول زمانبندی الگوریتم پیشنهادی و HEFT برابر و بهتر از Ahmad and Al Ebrahim به دست آمده است.

در گراف‌های با ۴۰۰ گره الگوریتم پیشنهادی با ۴۳۷ زمانبندی بهینه بیش از سایر الگوریتم‌ها به زمانیندی بهتر دست یافته است. این تعداد ۵۸,۲۷٪ کل تعداد آزمایشات است. در ۳۰۸ مورد یعنی ۴۱,۰۷٪، زمانبندی برابر با Ahmad and Al Ebrahim و بهتر از HEFT به دست آمده است. در ۵ مورد بقیه HEFT بهترین راه‌حل را ارایه کرده است.



شکل ۴-۳ – درصد تولید بهترین زمانبندی با ۲۰۰ وظیفه در الگوریتم‌های مختلف



شکل ۴-۴ – درصد تولید بهترین زمانبندی با ۴۰۰ وظیفه در الگوریتم‌های مختلف

ب) بهبود در طول زمانبندی

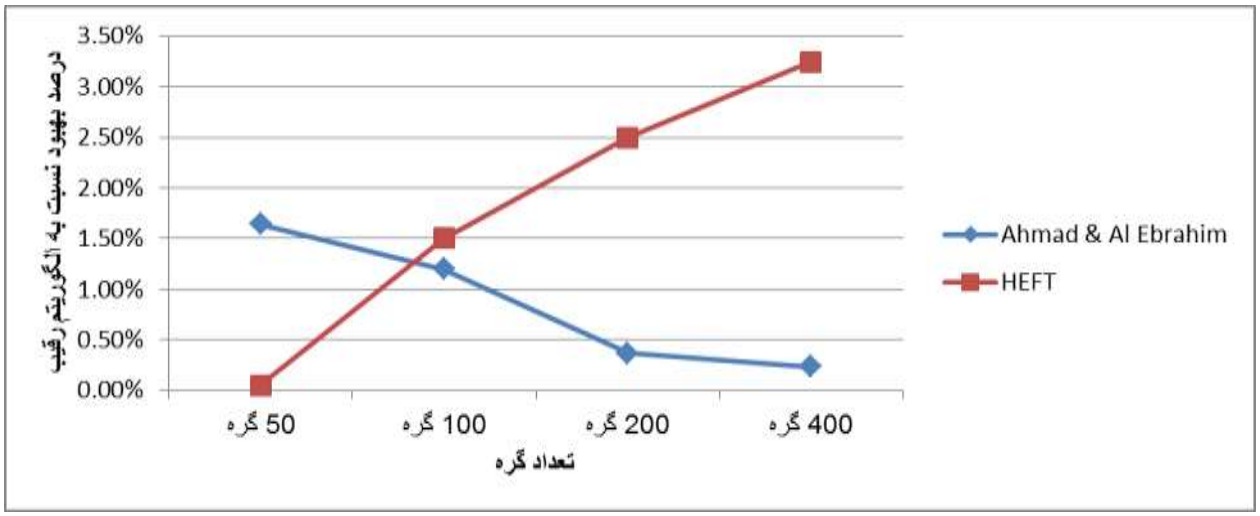
در این بخش به شرح میزان بهبود میانگین زمان اتمام کارها در زمانبندی هر الگوریتم در هر دوره از آزمایش پرداخته می‌شود. میزان بهبود نسبت به HEFT و Ahmad and Al Ebrahim به صورت جدا محاسبه شده است. در آزمایشات با گراف‌های ۵۰ گرهی جمع کل زمان اتمام کار برای HEFT برابر ۴۷۳۷۰۴، برای Ahmad and Al Eberahim ۴۸۱۳۳۸ و برای الگوریتم پیشنهادی ۴۷۳۴۲۹ به دست آمده است. با استفاده از فرمول میزان بهبود، بهبود نتایج نسبت به Ahmad and Al Ebrahim ۱,۶۴۳۱٪ و نسبت به HEFT ۰,۰۵۸۱٪ بوده است. این بهبود

در حالی است که HEFT نتایج بهینه‌تر بیشتری به نسبت الگوریتم پیشنهادی در آزمایشاتی با این تعداد از گره تولید کرده است. در گراف‌های با ۱۰۰ گره بهبود نسبت به HEFT ۱,۵۲۶۲٪ و نسبت به Ahmad and Al Ebrahim ۱,۲۲۶۶٪ بهبود در نتایج دیده شده است. هرچه تعداد گره بیشتر می‌شود، بهبود نسبت به HEFT بیشتر می‌شود. در آزمایش با ۲۰۰ گره نسبت به HEFT بهبود ۲,۵۳۶۲٪ و نسبت به Ahmad and Al Ebrahim بهبود ۰,۳۷۱۱٪ وجود دارد. در آزمایش با ۴۰۰ گره نیز نسبت به HEFT ۳,۲۴۷٪ و نسبت به الگوریتم پیشنهادی Ahmad and Al Ebrahim، ۰,۲۳۳٪ بهبود به دست آمده است.

جدول ۴-۱ - مجموع طول زمانبندی ها در تعداد وظایف مختلف

تعداد وظیفه	*الگوریتم پیشنهادی*	*HEFT*	*Ahmad et al*	*الگوریتم پیشنهادی بدون ورودی زمانبندی شده*
۵۰	۴۷۳۴۲۹	۴۷۳۷۰۴	۴۸۱۳۳۸	۵۰۹۰۰۹
۱۰۰	۸۷۸۸۲۸	۸۹۲۴۴۹	۸۸۹۷۴۲	۹۵۳۸۱۵
۲۰۰	۱۶۶۹۰۰۸	۱۷۱۲۴۴۰	۱۶۷۵۲۲۴	۱۸۵۲۰۳۳
۴۰۰	۳۲۴۷۳۰۶	۳۳۵۶۲۷۳	۳۲۵۴۸۸۹	۳۵۰۷۸۲۱

نمودار ۴-۴ بهبود در طول زمانبندی



نمودار ۴-۵ درصد بهبود میانگین طول زمانبندی در تعداد گره نسبت به HEFT و Ahmad et al

فصل ۵: بحث و نتیجه گیری و پیشنهادات

در این پژوهش به ارایه یک روش پیشنهادی جهت بهبود زمانبندی برای سیستم‌های پردازش موازی با چندین پردازنده ناهمگن پرداخته شده است، که در آن پردازنده‌ها جهت پردازش برنامه به وظایف مختلف تخصیص داده می‌شود و در انجام این کار باید به اولویت‌های انجام کار نیز توجه کرد. در الگوریتم پیشنهادی، به معرفی یک الگوریتم شبیه‌سازی تبرید ترکیب شده با یک الگوریتم بر پایه لیست پرداخته‌ایم که در آن یک زمانبندی تولید شده توسط الگوریتم بر پایه لیست به یک الگوریتم توسعه داده شده با استفاده از تکنیک شبیه‌سازی تبرید داده شده تا توسط آن در زمانبندی بهبود ایجاد شود. روش‌های مبتنی بر لیست وظایف معمولا دارای پیچیدگی زمانی کم هستند ولی زمانبندی ایجاد شده توسط آنها با زمانبندی بهینه فاصله دارد. در طرف دیگر زمانبندی توسط روش‌های جستجوی تصادفی نتایج بهینه‌تری را پیدا می‌کنند ولی به واسطه تعداد تکرار بسیار بالای آنها باعث ناکارآمدی این دسته می‌شود. با ترکیب این دو روش می‌توان جمعیت اولیه نزدیک به بهینه‌تری را وارد الگوریتم جستجوی تصادفی کرد و در نتیجه از تعداد تکرار ها کاسته و زمان دستیابی به نتیجه نزدیک به بهینه را کمینه کرد.

برای آزمایش الگوریتم پیشنهادی، چند الگوریتم مطرح دیگر به همراه الگوریتم پیشنهادی را با ۳۰۰۰ گراف با ۵۰، ۱۰۰، ۲۰۰ و ۳۰۰ گره اجرا کرده و در نهایت در ۱۴۵۰ مورد که شامل ۴۷,۵٪ کل آزمایشات می‌شود، بهتر از دیگر روش‌ها عمل کرده است.در کاهش طول زمانبندی نیز الگوریتم پیشنهادی به صورت میانگین بهبود ۰,۸۶٪ و ۱,۸۴٪ در طول زمانبندی داشته است.

در آینده می‌توان با تغییر در الگوریتم ورودی نتایج بهتر و متفاوت را بدست آورد. همچنین می‌توان به جای استفاده از شبیه‌سازی تبرید از تکنیک هایی مانند ژنتیک نیز بهره برد. با تغییر در آستانه و تعیین آستانه به نسبت مساله، می‌توان برای هر مساله شرایط بهتری فراهم کرد. مکانیزم تعیین برخلاف گراف‌های با تعداد بالای گره، در تعداد کم گره کارایی لازم را نداشته و معمولا باعث تاخیر در شروع و در نهایت افزایش طول زمانبندی می‌شود. در

قسمت بهبود نتایج می‌توان با افزودن پارامترهایی همچون میزان مصرف فضا و مصرف انرژی، از این روش برای استفاده در مسایل با محدودیت بیشتر استفاده کرد.

فهرست منابع غیر فارسی

1. Akbari M, Rashidi H, Alizade SH (2017) An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems, J Engineering Applications of Artificial Intelligence, (61):35-46
2. Al Ebrahim SH, Ahmad I (2016), Task scheduling for heterogeneous computing systems, J Supercomput
3. Arabnejad H, Barbosa JG (2014) List scheduling algorithm for heterogeneous systems by an optimistic cost table. IEEE Trans Parallel Distrib Syst 25(3):682–694
4. Arabnejad H (2013) List based task scheduling algorithms on heterogeneous systems-an overview. https://paginas.fe.up.pt/~prodei/dsie12/papers/paper_30.pdf. Accessed 5 Jun 2016
5. Arabnia HR, Oliver MA (1987) Arbitrary rotation of raster images with SIMD machine architectures. Int J Eurograph Assoc (Computer Graphics Forum) 6(1):3–12
6. Canon LC, Jeannot E, Sakellariou R, Zheng W (2008) Comparative evaluation of the robustness of dag scheduling heuristics. In: Grid Computing. Springer, US, pp 73–84
7. Dai Y, Zhang X (2014) A synthesized heuristic task scheduling algorithm. Sci World J 2014:1–9
8. Dhodhi MK, Ahmad I, Yatama A, Ahmad I (2002) An integrated technique for task matching and scheduling onto distributed heterogeneous computing systems. J Parallel Distrib Comput 62(9):1338– 1361
9. Hagras T, Janecek J (2003) A simple scheduling heuristic for heterogeneous computing environments. In: IEEE International Symposium on Parallel and Distributed Computing, pp 104–110
10. Herrmann J, Marchal L, Robert Y (2014) Memory-aware list scheduling for hybrid platforms. In: 2014 IEEE International In Parallel & Distributed Processing Symposium Workshops (IPDPSW), pp 689–698. doi:10.1109/IPDPSW.2014.80
11. Huang KC, Tsai YL, Liu HC (2015) Task ranking and allocation in list-based workflow scheduling on parallel computing platform. J Supercomput 71(1):217–240
12. Ilavarasan E, Thambidurai P, Mahilmannan R (2005) High performance task scheduling algorithm for heterogeneous computing system. In: Distributed and Parallel Computing. Springer Berlin Heidelberg, pp 193–203
13. Ilavarasan E, Thambidurai P (2007) Low complexity performance effective task scheduling algorithm for heterogeneous computing environments. J Comput Sci 3(2):94–103

14. Kwok YK, Ahmad I (1999) Static scheduling algorithms for allocating directed task graphs to multiprocessors. ACM Comput Surv 31(4):406–471
15. Shetti KR, Fahmy SA, Bretschneider T (2013) Optimization of the HEFT Algorithm for a CPU-GPU Environment. In: IEEE Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2013 International Conference on, pp 212–218
16. Topcuoglu H, Hariri S, Wu MY (2002) Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans Parallel Distrib Syst 13(3):260–274
17. Ullman JD (1975) NP-complete scheduling problems. J Comput Syst Sci 10(3):384–393
18. Wani MA, Arabnia HR (2003) Parallel edge-region-based segmentation algorithm targeted at reconfigurable multi-ring network. J Supercomput 25(1):43–63
19. Yang CH,Lee P, ChungYC(2007)Improving static task scheduling in heterogeneous and homogeneous computing systems. In: IEEE Parallel Processing, 2007. ICPP 2007. International Conference on, pp 45–45
20. Zhou H, Liu C (2014) Task mapping in heterogeneous embedded systems for fast completion time. In: ACM Proceedings of the 14th International Conference on Embedded Software, pp 1–10

Ministry of Science, Research and Technology

Seraj Higher Education Institute

Faculty of Engineering – Electricity and Computer Engineering Department

« M.Sc.»Thesis

On software engineering technology

Title :

Task Scheduling in Heterogeneous Systems using Simulated Annealing Algorithm

Supervisor :

S. Taghavi afshord(Ph.D.)

Advisor :

B. Arasteh (Ph.D.)

By :

M. Heidari Tarzam

September 2018