سیزدهمین کنفرانس ملی انجمن کامپیوتر ایران
جزیره کیش، خلیج فارس، ایران
۱۹ الی ۲۱ اسفند ۱۳۸۶
مقاله کامل
دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر
انجمن کامپیوتر ایران
Computer Society of Iran

زمان بندی فازی کار موازی در محیط ناهمگن

امیر کیوان شفیعی
گروه کامپیوتر ، دانشگاه فردوسی
مشهد، ایران
keivan2612@yahoo.com

حسین دلداری
دانشیار، گروه کامپیوتر، دانشگاه فردوسی
مشهد، ایران
hdeldari@yahoo.com

محمود نقیب زاده
استاد ، گروه کامپیوتر، دانشگاه فردوسی
مشهد، ایران
naghibzadeh@um.ac.ir

چکیده: ***زمان بندی وظیفه های موازی که به هم وابستگی داده ای دارند و یک کار را ایجاد می کنند یکی از مسایل پر کاربرد در زمینه علوم کامپیوتر ، حتی قبل از ظهور کامپیوترهای موازی بوده است. به خاطر NP-complete بودن مسأله هیچ راه حل زمان چند جمله ای در حالت کلی برای حل مسأله وجود ندارد بنابراین روش های اکتشافی زیادی برای حل این مسأله ارایه شده اند. با پیدایش محیطهای ناهمگن مثل کلاسترهای ناهمگن و گرید، پیچیدگی مسأله بخاطر ملاحظات ناهمگنی پردازنده ها (ماشینها) بخاطر وجود تفاوت قدرت پردازش آنها بیشتر شده است. ما در این مقاله روشی بنام SFW بر پایه یکی از روشهای موجود در این زمینه به نام زمان بندی لیست و منطق فازی پیشنهاد داده و با روش HEFT که یکی از روش های اکتشافی در این حیطه است، مورد مقایسه قرار داده ایم.***

واژه های کلیدی:زمان بندی وظیفه موازی، زمان بندی لیست، محیط ناهمگن، منطق فازی، NP-complete.

1- مقدمه

زمانبندی وظیفه[۱] های یک کار[۲] موازی یکی از مسایل مهم در زمینه پردازش موازی است که حتی قبل از ظهور ماشینهای موازی توجه بسیاری از محققان را به خود جلب کرده بود [۱]. مسأله زمانبندی به این شکل مطرح می شود که مجموعه ای از وظیفه ها که یک کار را تشکیل می دهند با یک گراف بدون دور جهت دار[۳] (DAG) مدل سازی می شود که در آن یالها و راسهای آن دارای وزن می باشند و در کنار آن مجموعه ای از پردازنده ها وجود دارد که ممکن است همسان یا ناهمسان باشد. هدف قرار دادن وظیفه ها روی پردازنده ها به گونه ای است که زمان کلی اتمام کار حداقل گردد.

مسأله در شکل عمومی خود یک مسأله NP-complete است [۹،۱۰،۷] و راه حلهای زمان چند جمله ای[۴] تنها برای موارد محدودی شناخته شده اند[۱۱]. بنابراین تلاشهای بسیاری برای ارایه الگوریتمهای اکتشافی[۵] در این زمینه انجام شده است که منجر به ارایه هزاران الگوریتم گردیده است.

می توان الگوریتمهای زمان بندی را به ۵ دسته الگوریتم تقسیم بندی کرد[۳،۵]: الگوریتمهای زمان بندی لیست[۶]، زمان بندی خوشه بندی[۷]، زمان بندی تکرار وظیفه ها[۸]، روشهای جستجوی هدایت شده[۹] و روشهای ترکیبی.

در زمانبندی لیست که یکی از محبوبترین روشها می باشد به این صورت عمل می شود که به وظیفه ها اولویتی براساس روشهایی واگذار می شود به نحوی که قوانین از پیش تعیین شده ای را برآورده سازد، سپس وظیفه ها را براساس اولویت واگذار شده در یک لیست به نام لیست اولویت یا آماده قرار می دهد اگر این لیست در حین واگذاری وظیفه ها به پردازنده ها تغییر نکند به آن لیست ایستا[۱۰] و در غیر اینصورت به آن لیست پویا[۱۱] می گویند. پس از ایجاد لیست اولویت، وظیفه ها به پردازنده ها به نحوی واگذار می شود که یک تابع هزینه ای[۱۲] مناسب را مینیمم کند. روش های اکتشافی زمانبندی لیست بر روی چگونگی اولویت دهی به وظیفه ها و همچنین تا حدودی بر روی تابع هزینه برای واگذاری وظیفه ها به پردازنده ها تمرکز دارند.

مسأله دیگر در زمانبندی وظیفه، محیط کاری آن است که می توان به دو دسته تقسیم کرد: زمانی که پردازنده ها دقیقاً یکسان باشند که به آن محیط همگن[۱۳] می گویند ، و زمانی که پردازنده ها دارای قابلیتهای متفاوتی هستند که به آن محیط ناهمگن می گویند. در محیط ناهمگن[۱۴] چالش جدیدی که ایجاد می شود این است که در زمان واگذاری وظیفه ها به پردازنده ها باید به قابلیت پردازنده ها توجه کرد. این قابلیتها بیشتر در تفاوت اجرای وظیفه ها در پردازنده های متفاوت می باشد.

در محیط ناهمگن چون ممکن است یک وظیفه تنها روی تعداد معدودی پردازنده قابلیت اجرایی داشته باشد، نیاز به بررسی بیشتری برای رسیدن به هدف کمترین زمان اجرا وجود دارد؛ در بخش ۲ به این موضوع باجزئیات بیشتری خواهیم پرداخت.

مسایلی که در محیط نا همگن مطرح می شوند، چارچوب مناسبی برای استفاده از منطق فازی برای تاثیر اولویت دادن به پردازنده ها را فراهم می نماید. در این مورد در بخش ۴ سخن گفته خواهد شد.

در این مقاله ما با مطرح کردن مسایل مرتبط با زمان بندی وظیفه روشی بر اساس منطق فازی به نام زمان بندی با وزن دهی فازی (SFW)[۱۵] برای رسیدن به هدف مینیمم کردن زمان اجرای یک کار را ارائه می دهیم و سپس نتایج حاصل از این روش را با یکی از روشها به نام HEFT [۵] در این زمینه مورد مقایسه قرار خواهیم داد.

این مقاله به صورت زیر سازماندهی شده است: در بخش ۲ به تعریف مسأله می پردازیم، بخش ۳ مرتبط است با کارهای انجام شده قبلی در این زمینه ؛ و بخش ۴ به ارائه الگوریتم پیشنهادی می پردازد. در بخش ۵ به مقایسه نتایج می پردازیم و سرانجام بخش ۶ مربوط است به نتیجه گیری.

2- تعریف مسأله

فرض کنید که n وظیفه وابسته به هم که یک کاربرد[۱۶] را ایجاد می کنند وجود دارند ، می توان این n وظیفه را به صورت یک گراف بدون دور جهت دار (DAG) مدل سازی کرد به صورتی که هر گره وظیفه مورد نظر را نشان دهد و در صورت وابستگی داده ای بین این وظیفه ها یالی از وظیفه تولید کننده داده به وظیفه مصرف کننده داده ایجاد می شود. هر گره وزن (یا ونهایی) دارد که مشخص کننده زمان مورد نیاز برای انتقال داده ها از یک ماشین به ماشین دیگر را مشخص می کند. همچنین p پردازنده وجود دارند که به عنوان محیط کاری مطرح می شوند. مسأله چگونگی قرار دادن این n وظیفه بر روی p پردازنده است به گونه ای که زمان اتمام کل کاربرد (makespan) [۲۰] مینیمم گردد.

اگر $G = (V, E)$ یک گراف وظیفه[۱۷] باشد؛ V مجموعه راسهای آن است که وظیفه ها را نشان می دهد. هر وظیفه رشته ای از دستورالعملها است که به طور سریال روی یک ماشین (پردازنده) اجرا شود. E مجموعه یالهای G است. یال $e_{i,j}$ گره v_i را به گره v_j متصل می کند و بدین صورت v_i والد v_j و v_j فرزند v_i است. همچنین وجود یال $e_{i,j}$ دلالت بر این دارد که قبل از شروع v_j باید v_i تمام شده باشد و داده های آن بر روی پردازنده ای که v_j قرار است در آن اجرا شود ، قرار گرفته باشد.

وظیفه بدون هیچ والدی وظیفه ورودی[۱۸] و وظیفه بدون هیچ فرزندی وظیفه خروجی[۱۹] نامیده می شوند. بدون اینکه به کلیت مسأله خدشه ای وارد شود می توان در صورتی که یک DAG بیش از یک گره ورودی داشته باشد با استفاده از یک وظیفه با وزن صفر و هزینه ارتباطی به تمام وظیفه های ورودی ، یک DAG با تنها یک وظیفه ورودی ایجاد کرد. همچنین کار مشابهی برای گره های خروجی انجام می دهیم تا یک DAG با تنها یک گره ورودی و یک گره خروجی داشته باشیم.

مدل محیط محاسباتی یک محیط ناهمگن متشکل از p پردازنده است که دارای توپولوژی ارتباطی کاملاً متصل است و فرض می شود:

- هر ماشین (پردازنده) اجرای وظیفه و ارتباط با دیگر ماشینها را همزمان انجام می دهد؛
- اجرای وظیفه ها روی ماشینها بدون وقفه انجام می گیرد
- هزینه اراتباطی دو وظیفه که در یک ماشین اجرا می شوند صفر در نظر گرفته می شود

همانطور که گفته شد مدل دارای یک DAG است که یالهای آن وزن دارند. وزن یالها با ماتریس E نمایش داده می شود؛ که در آن $e_{i,j}$ نشان دهنده مقدار داده های انتقالی از وظیفه v_i را به وظیفه v_j است تا بتواند v_j اجرا شود و بر حسب بایت بیان می شود. علاوه براین یک ماتریسW وجود دارد که ابعاد آن $p \times n$ است. که n تعداد راسهای گراف یا وظیفه ها می باشد و p تعداد پردازنده ها است. $w_{i,j}$ نشان دهنده زمان اجرای وظیفه v_i بر روی پردازنده p_j است. با توجه به ناهمگن بودن محیط عملیاتی ممکن است وظیفهی مثل v_i بر روی پردازنده های مختلف زمانهای اجرایی متفاوتی داشته باشند. یک ماتریسdtr با ابعاد $p \times p$ وجود دارد که $d_{i,j}$ مشخص کننده نرخ انتقال داده ها از پردازنده p_i به p_j می باشد. با توجه به مفروضات که قبلاً عنوان شده است قطر اصلی dtr صفر است.

زمان مورد نیاز برای انتقال داده از پردازنده p_m به پردازنده p_n زمانی که داده ها از v_i در p_m می خواهند به p_n که v_j در آن قرار است اجرا شود ، منتقل شود ($c_{i|m,j|n}$)به صورت (۱) است.

$$c_{i|m,j|n}=d_{m,n}.e_{i,j} \qquad (۱)$$

که درآن $d_{m,n}$ نرخ انتقال داده ها از پردازنده p_m به پردازنده p_n و $e_{i,j}$ مقدار داده ای است که برای اجرا v_j از v_i باید دریافت کند.

در شکل ۱ نمونه ای از یک DAG به همراه ماتریسهای وزن و نرخ ارتباطات نمایش داده شده اند.

هدف اصلی در زمانبندی ، انتخاب زمانبندی مناسب بر روی پردازنده ها به گونه ای است که زمان اتمام کلی کاربرد مینیمم گردد

برای ادامه بحث به ارایه چند تعریف می پردازیم که در ادامه به آنها مراجعه خواهیم نمود:

تعریف ۱:

مسیر بحرانی:طولانیترین مسیر از گره ورودی به گره خروجی با احتساب وزن گره ها و وزن یالها.

تعریف ۲:

$t\text{-level}(n_i)$: طولانی ترین مسیر از یک گره ورودی به گره n_i.

تعریف ۳:

$b\text{-level}(n_i)$: طول طولانی ترین مسیر از گره n_i به یک گره خروجی.

در تعاریف ۲ و ۳ منظور از طول مسیر مجموع وزن تمام گره ها و یالها در مسیر است.

3- کارهای انجام شده

الگوریتمهای زیادی برای زمان بندی در نوشتجات پیشنهاد شده اند. در [۴] دسته بندی و مقایسه بین روشهای موجود برای زمان بندی DAG در محیط همگن ارایه شده است. الگوریتمهای مختلفی برای زمان بندی لیست ارائه شده اند که در زیر به چند نمونه که برای محیطهای نا همگن ارایه شده اند می پردازیم:

مسیر بحرانی اصلاح شده[۲۰] (MCP): الگوریتم MCP [۷] اولویت یک وظیفه را بر اساس مقدار " در دیرترین زمان ممکن گره اش(ALAP)" واگذار می کند. زمان ALAP یک گره با رابطه زیر تعریف می شود:

$$ALAP(n_i) = T_{critical} - b\_level(n_i)$$

که $T_{critical}$ طول مسیر بحرانی است. ALAP یک گره معیاری از مدت زمانی است که زمان شروع وظیفه می تواند به تاخیر بیافتد بدون اینکه طول زمانبندی افزایش یابد. MCP ابتدا ALAP تمام گره ها را محاسبه می کند؛ سپس گره ها را براساس زمان ALAP آنها به ترتیب صعودی در لیست قرار می دهد. گره ها با استفاده از کمترین زمان ALAP گره های بعدی شکسته می شوند، و این کار تا آخر ادامه می یابد. سپس گره ابتدای لیست را به پردازنده ای که کمترین زمان شروع را می دهد، واگذار می کند. پس از آن گره زمان بندی شده از لیست خارج می شود. این گام تا زمانی که لیست خالی شود ، ادامه می یابد. پیچیدگی زمانی MCP $O(n^2 \log n)$ است.

زمانبندی سطح پویا[۲۱] (DLS): DLS [۸] یکی از ابتدایی ترین الگوریتمهای زمانبندی لیست است که برای سیستم محاسباتی ناهمگن طراحی شده است. در مقایسه با زمانبندی لیست معمولی، DLS یک لیست زمانبندی را در حین فرآیند زمانبندی نگهداری نمی کند. DLS اولویتهای گره را به طور پویا با واگذاری یک ویژگی به نام سطح پویا[۲۲] (DL) به همه گره های زمانبندی نشده در هر گام زمانبندی مشخص می کند. DL وظیفه n_i روی p_j که با $DL(n_i, p_j)$ مشخص می شود مقدار تطبیق n_i را روی p_j مشخص می کند. $DL(n_i, p_j)$ با فرمول زیر مشخص می شود:

$$DL(n_i, p_j) = SL(n_i) - ST(n_i, p_j)$$

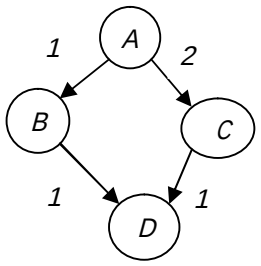
که در آن $SL(n_i)$، b-level ایستای n_i است که همان b-level است که شامل هزینه ارتباطی مسیر نمی باشد.

$ST(n_i, p_j)$ زمان شروع n_i را در p_j مشخص می کند. در هر گام زمانبندی DLS ، DL را برای هر یک از گره های آماده روی هر یک از پردازنده ها مشخص می کند.سپس جفت پردازنده – گرهی که بزرگترین DL در میان همه جفتها دارد را انتخاب می نماید. این فرآیند تا زمانبندی تمام گره ها ادامه می یابد.

مسیر بحرانی پویا[۲۳] (DCP): در [۶] ، Kwork و دیگران ، روش های اکتشافی برای زمانبندی لیست پیشنهاد کرده اند که شامل ویژگی های زیر است:

بجای نگهداری یک لیست زمانبندی ایستا، الگوریتم گره بعدی را برای زمانبندی به طور پویا انتخاب می کند. DCP اولویتها را برای گره ها در هر گام بر پایه مسیر بحرانی پویا[۲۴] (DCP) چنان انتخاب می کند که طول زمانبندی می تواند به طور یکنواخت کاهش یابد. به منظور کاهش طول DCP در هر گام زمانبندی انتخاب می شود که هیچ گره والدی زمانبندی نشده برای آن وجود ندارد.

در حین فرآیند گام انتخاب، الگوریتم یک زمان شروع پیش بینی شده را استفاده می کند. فرض کنید n_i برای زمانبندی بررسی می شود. فرض کنید n_c فرزند گره n_i باشدکه کمترین تفاوت بین حد بالا و حد پایین زمان شروع را دارد. سپس n_i باید در پردازنده ای زمانبندی شود که جمع حدهای بالای زمان شروع n_i و n_c را مینیمم کند.



(a)

	p1	p2
A	3	2
B	∞	1
C	2	4
D	1	1

(b)

	p1	p2
P1	1	0
P2	1	0

(c)

شکل 1 (a) نمونه ای از یک DAG (b) زمان اجرای هر تسک در دو پردازنده p1 و p2 (c)نرخ انتقال داده بین پردازنده ها

کمترین زمان اتمام ناهمگن[۲۵] (HEFT): در [۵] ، Topcouglu و دیگران روشی اکتشافی به نام الگوریتم HEFT ارائه داده اند. الگوریتم HEFT وزن یک گره را هزینه اجرای میانگین گره در میان تمام پردازنده های موجود قرار می دهد. متشابهاً ، وزن یک یال، میانگین هزینه ارتباطی میان همه لینکها به تمام ترکیبات جفت پردازنده های متصل، محاسبه می شود. اولویت یک وظیفه b-level آن گره است که بر پایه میانگین ارتباطات انتخاب می شود.لیست وظیفه با مرتب سازی وظیفه ها به ترتیب کاهش b-level ایجاد می شود. وظیفه انتخاب شده به پردازنده ای اختصاص می یابد که کمترین زمان اتمام را دارد.

SDC[۲۶] :در SDC [۳] الگوریتم در دو فاز زمانبندی لیست به صورت زیر عمل می کند:

- ابتدا بر اساس قابلیت پردازنده ها وزنها تنظیم می شوند به گونه ای که وزن بیشتر به پردازنده ای تعلق می گیرد که در تعداد

کمتری پردازنده قابلیت اجرا داشته باشد و سپس با استفاده از وزنهایی که بدست می آید و با استفاده از b-level وزنها را در لیست قرار می دهد.

- در فاز دوم الگوریتم بر اساس فاکتور اجرای وظیفه های فرزند روی پردازنده ها و EFT، پردازنده ی مناسب برای اجرا انتخاب می شود.

برخی دیگر از الگوریتمهای زمانبندی بر اساس زمان بندی لیست شامل: روش اکتشافی نگاشت[۲۷] (MH) [۱۲] ،مسیر بحرانی سریع[۲۸] (FCP) [۱۳] و روش اکتشافی زمانبندی درجی[۲۹] (ISH) [۱۶] می باشند.

4- شرح الگوریتم پیشنهادی

در این قسمت به شرح الگوریتم پیشنهادی برای حل مسأله زمانبندی DAG می پردازیم.

ما از روش زمان بندی لیست استفاده می کنیم که در الگوریتم ۱ نشان داده شده است.

در زمان بندی لیست ۲ فاز وجود دارد: (۱) محاسبه اولویت گره ها و (۲) تعریف تابع هزینه. در ابتدا باید برای هر یک از گره ها اولویتی حساب کرد. دو روش پر استفاده برای این منظور b-level و t-level است.

ابتدا برای فازهای مختلف توضیحی ارایه می دهیم سپس الگوریتم را به صورت شبه کد بیان می نماییم.

4-1- تنظیم وزنها

در این قسمت مطالبی در رابطه با تنظیم وزن گره ها و یالهای DAG بیان می کنیم. این کار برای استفاده از b-level که برای اولویت دادن به گره ها استفاده می شود لازم است زیرا که b-level از تنها یک مقدار استفاده می کند و این در حالی است که در محیط ناهمگن مقادیر برای گره ها و یالها متفاوت است.

ابتدا وزن یالها را به صورت زیر تعریف می کنیم:

$$c_{i,j}=e_{i,j}.\ \overline{dtr} \qquad (۲)$$

که در آن $e_{i,j}$ مقدار داده ای است که از v_i به v_j برای اجرا منتقل می شود و $\overline{dtr}$ میانگین نرخ انتقال داده بین پردازنده ها است.

الگوریتم 1 زمان بندی لیست

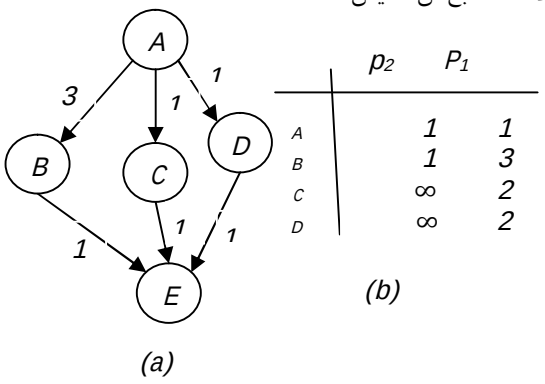
```
Calculate the priority of each task according to some predefined formula
PriorityList = {v1, v2, ..., vn} is sorted by descending order of task priorities
while PriorityList is not empty
    Remove the first task from the PriorityList and assign it to an appropriate processor in order to optimize a predefined cost function
return (schedule)
```

روشهای مختلفی برای تنظیم وزنهای یک گره در محیط ناهمگن وجود دارد[۲] [۳] . از جمله این روشها می توان به مقادیر میانگین، بهترین مقدار و غیره اشاره کرد. در این الگوریتم ما با استفاده از معیارهای وزن گره ، قابلیت پردازنده ها ، و تعداد فرزندان و با استفاده از روش منطق فازی برای همه گره های DAG یک وزن بدست می آوریم. در زیر این معیارها بررسی شده اند:

معیار ۱ (وزن گره ها): گره ای با وزن بیشتر دارای وزن خروجی بیشتری نسبت به گره ای با وزن کمتر است.

معیار ۲ (قابلیت پردازنده ها): به گرهی که دارای پردازنده های کمتری برای اجرا است باید وزن بیشتری تعلق گیرد تا از وضعیتهایی که تاثیرات نامطلوب دارند اجتناب شود [۳].

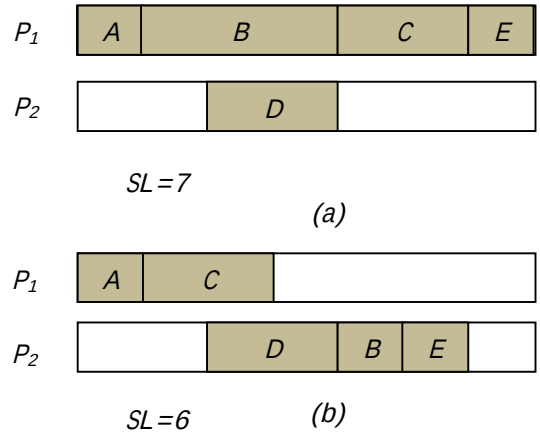
برای شرح معیار ۲ به یک مثال و تاثیر اولویت دادن به وظیفه ها برروی زمانبندی می پردازیم. در شکل ۲ یک نمونه گراف وظیفه به همراه اطلاعات منابع آن نمایش داده شده است.



شکل 2 یک DAG به همراه اطلاعات منابع

همانطور که مشهود است میانگین زمان اجرای B ، C و D همگی ۲ می باشد بنابراین در روشهایی که از میانگین وزنها استفاده می کنند، ترتیب زمانبندی یکسان می باشد. ولی در شکل۳ مشهود است که انتخاب وظیفه ای که پردازنده کمتری دارد بر روی زمانبندی تأثیر بهتری ایجاد می نماید. در قسمت (a) ابتدا B زمان بندی می شود ولی

در قسمت (b) ابتدا C زمانبندی می شود وبدین خاطر چون انتخاب با وظیفه ای است که پردازنده های کمتری در اختیار دارد ، اجبار کمتری بر روی آن قرار می گیرد و بنابراین زمانبندی بهتری حاصل می گردد.
با توجه به مثال مذکور مشهود است که در زمانی که اولویت به وظیفه ای تعلق می گیرد که دارای پردازنده های کمتری است نتایج بهتری حاصل می شود. در سیستم فازی ما اولویت با قرار دادن وزن بیشتر بر روی گره هایی که دارای پردازنده های کمتری برای اجرا هستند، بدست می آوریم.
معیار ۳ (فرزندان گره):هر گرهی که فرزندان بیشتری دارد، دارای اولویت بیشتری برای زمان بندی است.
معیار ۳ از این جهت مهم است که با وجود فرزندان بیشتر برای گره نسبت به گره های دیگر که فرزندان کمتری دارند، وزن بیشتری باید تعلق گیرد.
در ابتدا فازی سازی[۳۰] ورودی بر اساس توابع عضویت برای هر یک از ورودی ها بر اساس شکل ۴ انجام می گیرد. همانطور که می بینید سیستم ما دارای ۳ ورودی و ۱ خروجی است و برای تمام آنها از توابع مثلثی برای فازی سازی استفاده شده است.



شکل 3دو نمونه زمان بندی برای گراف نمونه شکل 3 (a) بدون در نظر گرفتن قابلیتهای پردازنده ها (b) با در نظر گرفتن قابلیت پردازنده ها

پایگاه قوانین مورد استفاده در این کار در جدول ۱ آورده شده است که هر سطر جدول نشانگر یک قانون می باشد. برای مثال سطر اول مشخص کننده قانون زیر است:
If nw is ZO and cp is ZO and nc is ZO then w is ZO
ما برای این پایگاه قوانین از موتور استنتاج[۳۱] Mamdani و از غیر فازی ساز[۳۲] centroid استفاده کرده ایم[۱۸].

4-2- اولویت دادن به گره ها

این گام برای زمانبندی لیست لازم است. یک لیست پردازش وظیفه توسط مرتب سازی کاهشی براساس برخی از توابع درجه[۳۳] از قبل تعیین شده ساخته می شود. در این مقاله، ما از b-level به عنوان تابع درجه استفاده می کنیم[۵]که به صورت بازگشتی در رابطه (۳) تعریف شده است.

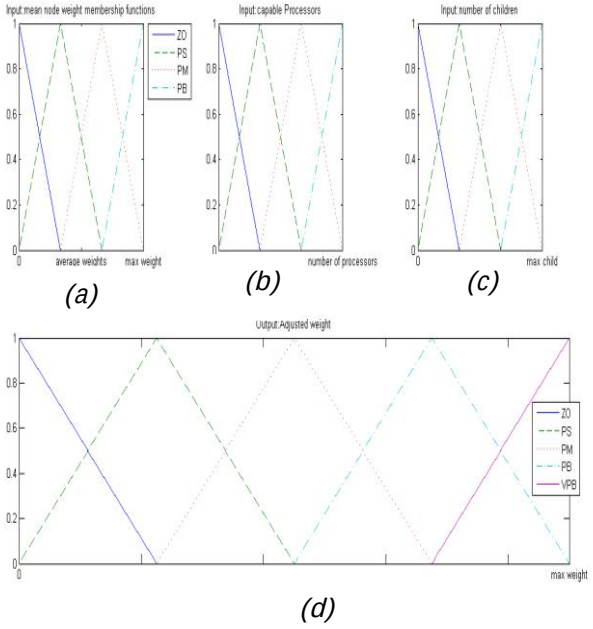
$$BLEV(v_i) = w_i + \max_{v_j \in succ(v_i)}\{\overline{c_{i,j}} + BLEV(v_j)\} \qquad (۳)$$

که در آن $\overline{c_{i,j}}$ هزینه ارتباط میانگین $e_{i,j}$، w_i وزن گره v_i و $succ(v_i)$ مجموعه تمام گره های بعدی (فرزندان بلافصل) v_i است.

4-3- انتخاب پردازنده ها

معیارهای متفاوتی برای انتخاب پردازنده مناسب برای یک وظیفه پیشنهاد شده است. در محیط همگن EST انتخاب محبوبتری است [۱۴] [۱۵] [۱۷]. در (۴) فرمول بدست آوردن EST را بیان می کند.

$$EST(v_i,p_j) = \max\{avail(v_i,p_j), \max_{v_k \in pred(v_i)}(FT(v_k,p_{sk}) + c_{k|sk,i|j})\} \qquad (۴)$$



شکل 4 توابع عضویت (a) وزن گره ها (b) قابلیت پردازنده ها (c) تعداد فرزندان (d) خروجی

1 node weight,2cappabilities processors, 3 number of childrens, 4 weight

nw[1]	cp[2]	nc[3]	w[4]	nw	cp	w	nw
ZO	ZO	ZO	ZO	PM	ZO	ZO	PM
ZO	ZO	PS	PS	PM	ZO	PS	PM
ZO	ZO	PM	PM	PM	ZO	PM	PB
ZO	ZO	PB	PM	PM	ZO	PB	PB
ZO	PS	ZO	ZO	PM	PS	ZO	PM
ZO	PS	PS	PM	PM	PS	PS	PM
ZO	PS	PM	PM	PM	PS	PM	PM
ZO	PS	PB	PS	PM	PS	PB	PB
ZO	PM	ZO	ZO	PM	PM	ZO	PM
ZO	PM	PS	ZO	PM	PM	PS	PM
ZO	PM	PM	ZO	PM	PM	PM	PM
ZO	PM	PB	PS	PM	PM	PB	PM
ZO	PB	ZO	ZO	PM	PB	ZO	PB
ZO	PB	PS	ZO	PM	PB	PS	VPB
ZO	PB	PM	ZO	PM	PB	PM	VPB
ZO	PB	PB	ZO	PM	PB	PB	VPB
PS	ZO	ZO	PS	PB	ZO	ZO	PB
PS	ZO	PS	PM	PB	ZO	PS	PB
PS	ZO	PM	PB	PB	ZO	PM	VPB
PS	ZO	PB	PB	PB	ZO	PB	VPB
PS	PS	ZO	PS	PB	PS	ZO	PB
PS	PS	PS	PS	PB	PS	PS	PB
PS	PS	PM	PM	PB	PS	PM	PB
PS	PS	PB	PB	PB	PS	PB	VPB
PS	PM	ZO	PS	PB	PM	ZO	PB
PS	PM	PS	PS	PB	PM	PS	PB
PS	PM	PM	PS	PB	PM	PM	PB
PS	PM	PB	PM	PB	PM	PB	VPB
PS	PB	ZO	PM	PB	PB	ZO	PB
PS	PB	PS	PB	PB	PB	PS	PB
PS	PB	PM	VPB	PB	PB	PM	PB
PS	PB	PB	VPB	PB	PB	PB	PB

که درآن $avail(v_i, p_j)$ نزدیکترین زمانی است که v_i می تواند در p_j اجرا شود، $pred(v_i)$ مجموعه ای از والد های بلافصل v_i است. درونی ترین max زمانی را محاسبه می کند که تمام داده ها برای اجرای v_i در پردازنده p_j ، به عبارت دیگر زمان آماده[۳۴]، موجود باشند. این زمان با در نظر گرفتن تمام والدهای v_i زمانی که آنها تمام می شوند (FT[۳۵]) و زمان مورد نیاز برای انتقال داده ها به پردازنده ای که v_i در آن اجرا می شود حاصل می گردد.

در محیط ناهمگن استفاده از EFT ، زمانبندی بهتری را موجب می شود.[۵] EFT به صورت زیر تعریف می شود:

$$EFT(v_i, p_j) = w_{i,j} + EST(v_i, p_j) \qquad (۵)$$

که در آن $w_{i,j}$ وزن v_i روی p_j است.

$$L = FT(n_{exit}) \qquad (۶)$$

ما در این مقاله از EFT استفاده می کنیم که در (۵) تعریف شده است.

طول زمان بندی با (۶) تعریف می شود:

هر چند معیارهای کارآیی مختلفی مثل تأخیر ورود[۳۶] یا زمان جریان کلی[۳۷] در نوشتجات معرفی شده اند[۱۹]، هدف ما مینیمم کردن L در (۶) می باشد.

4-4- طرز عمل الگوریتم

یک شبه کد[۳۸] الگوریتمSFW در الگوریتم ۲ نمایش داده شده است.

الگوریتم 2 الگوریتم پیشنهادی (SFW)

Set weights of tasks node with Fuzzy Weight assigning (sec. 1-4)
Set weight of edges with Eq. (2)
Compute b-levels for all tasks by traversing graph upward from the exit node
Sort the task into a list by non-increasing order of b-level
While** the scheduling list is not empty **do
  Remove the first task v_i from the list for scheduling
  For each** processor capable p_j of v_i **do
    Compute EFT(v_i, p_j) with Eq. (5) using insertion based policy
  End for
  Assign task v_i to processor that minimize of v_i
End while

5- نتایج تجربی

در این بخش ، ما نتایج تطبیقی SFW را با HEFT نشان می دهیم. ابتدا معیارهای مقایسه را توضیح می دهیم سپس نتایج را بررسی می کنیم.

5-1- معیارهای مقایسه[۳۹]

مقایسه دو الگوریتم با دو معیار زیر انجام می شود:

طول زمانبندی نرمال شده[۴۰] (NSL):

معیار عمده کارآیی یک الگوریتم طول زمانبندی خروجی اش است. NSL یک الگوریتم به صورت زیر تعریف می شود:

$$NSL = \frac{L}{\sum_{v_i \in cp_{min}} \min_{p_j \in P}\{w_{i,j}\}} \qquad (۷)$$

که در آن L طول زمانبندی است. cp_{min} مسیر بحرانی DAG است وقتی که هزینه محاسباتی، مینیمم وزنهای گره های مسیر بحرانی در میان پردازنده های قابل استفاده[۴۱] ارزیابی می شود؛ به عبارت دیگر برای محاسبه cp_{min} باید گره های مسیر بحرانی را پیدا کرد وسپس برای هر گره ، وزن بر روی پردازنده ای انتخاب می شود که آن وظیفه را سریعتر اجرا می کند. مخرج (۷) یک حد پایین روی طول زمان بندی را نشان می دهد. این حد پایین هیچگاه قابل دسترس نیست[۳] و برای هر الگوریتمی $NSL \geq 1$ می باشد. ما از NSL روی مجموعه ای از DAG ها به عنوان یک متریک مقایسه ای استفاده می کنیم.

میانگین درصد تنزل[۴۲] (APD):

APD یک الگوریتم ، درصد تنزل طول زمان بندی L ایجاد شده توسط الگوریتم نسبت به بهترین زمان بندها است. فرض کنید G مجموعه ای از DAG ها باشد،یعنی $G = \{g_1, g_2, \dots\}$ و $ALG = \{alg_1, alg_2, \dots\}$ مجموعه ای از الگوریتمها است که ما با هم مقایسه می کنیم. $sl(alg_i, g_j)$ طول g_j را با استفاده از الگوریتم alg_i نشان می دهد. APD الگوریتم alg_i روی مجموعه گرافهای G به صورت (۸) تعریف می شود:

$$APD(alg_i, G) = \frac{\sum_{g_j \in G}(sl(alg_i, g_j) - sl(\underset{alg \in ALG}{argmin} sl(alg, g_j)), g_j))}{\|G\|} \qquad (۸)$$

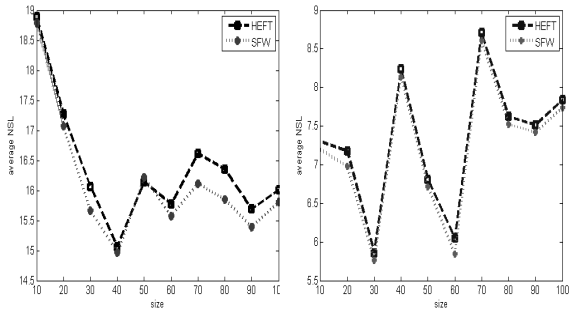
نرخ ارتباطات به محاسبات[۴۳] (CCR):

نرخ ارتباطات به محاسبات که به صورت میانگین مجموع وزن یالهای گراف به میانگین وزن راسهای گراف است به عنوان معیاری برای مقایسه مرسوم است. ما در دو حالت CCR<1 و CCR>=1 به بررسی نتایج می پردازیم.

5-2- مشاهده و تحلیل نتایج

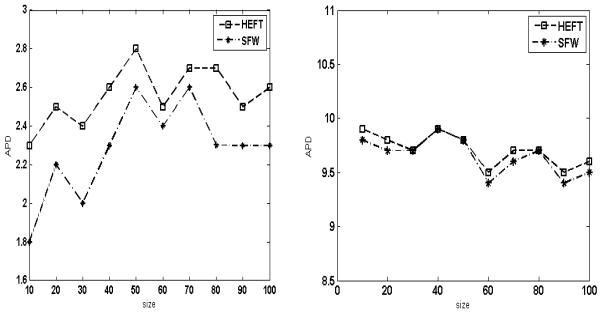
در شکل ۶ و ۷ تاثیر روش پیشنهادی بر روی NSL و APD نشان داده شده اند. برای هریک از معیارهای مقایسه از دو نمودار برای حالتهای CCR>=1 و CCR<1 نمایش داده شده اند.

همانطور که مشهود است در حالت CCR<1 نسبت به CCR>=1 الگوریتم پیشنهادی ما بهتر عمل می کند. این بخاطر این موضوع است که ما از b-level استفاده می کنیم و با CCR زیاد تاثیر میانگین ارتباطات بیشتر می شود و بدین لحاظ وزن گره ها بر روی زمان بندی تاثیر چندانی نخواهد داشت. از سوی دیگر در CCR<1 چون وزن بر روی زمان بندی تاثیر بیشتری دارد بنابراین روش ما بهتر عمل خواهد کرد.



(a) (b)

شکل 5 مقایسه HFET و SFW با معیار NSL میانگین در (a) CCR<1 (b) CCR>=1



(a) (b)

شکل 6 مقایسه HEFT وSFW با معیار APD در (a) CCR<1 (b) CCR>=1

6- نتیجه گیری

در این مقاله ما الگوریتم جدیدی برای زمان بندی کاربردهایی بر پایه DAG در سیستم ناهمگن ارایه داده ایم که در آن پردازنده ها دارای قابلیتهای متفاوتی می باشند. الگوریتم با استفاده از منطق فازی وزنی برای وظیفه ها تعیین می نماید تا با این کار بتوان از b-level جهت ساخت لیست آماده سود برد. فاکتورهای تصمیم گیری برای سیستم فازی از سه پارامتر بدیهی تشکیل شده است که شامل میانگین وزن گره، تعداد پردازنده هایی که در آنها قابلیت اجرا دارد، و تعداد فرزندان آن گره می باشد. نتایج نشان می دهد که SWF نسبت به HEFT در جایی که محاسبات نسبت به ارتباطات بیشتر است، بهتر عمل می کند.

مراجع

[1] Y.-K. Kwok and I. Ahmad. Benchmarking and comparison of the task graph scheduling algorithms. J. Parallel Distrib. Comput., 59(3):381–422, 1999.
[2] H. Zhao and R. Sakellariou. An experimental investigation into the rank function of the heterogeneous earliest finish time scheduling algorithm. In Proceedings of 9th International Euro-Par Conference. Springer-Verlag, 2003.
[3] Zhiao Shi. Scheduling tasks with precedence constraints on heterogeneous distributed computing systems. PhD thesis University of Tennessee, Knoxville, 2006
[4] Y.-K. Kwok and I. Ahmad. Static scheduling algorithms for allocating directed task graphs to multiprocessors. ACM Comput. Surv., 31(4):406–471, 1999.
[5] H. Topcuoglu, S. Hariri, and M.-Y. Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans. Parallel Distrib. Syst., 13(3):260–274, 2002.
[6] Y.-K. Kwok and I. Ahmad. Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors. IEEE Trans. Parallel Distrib. Syst., 7(5):506–521, 1996.
[7] Dandass, Y. S. Genetic List Scheduling for Soft Real-Time Parallel Applications. IEEE Congress on Evolutionary Computation. June 2004, 1164-1171.
[8] G. C. Sih and E. A. Lee. A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures. IEEE Trans. Parallel Distrib. Syst., 4(2):175–187, 1993.
[9] H. El-Rewini, T. G. Lewis, and H. H. Ali. Task Scheduling in Parallel and Distributed Systems. Prentice Hall, Englewood Cliffs, 1994.
[10] M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA, 1979.
[11] E.G. Coffman, Computer and Job-Shop Scheduling Theory, Wiley, New York, 1976.
[12] H. El-Rewini and T. G. Lewis. Scheduling parallel program tasks onto arbitrary target machines. J. Parallel Distrib. Comput., 9(2):138–153, 1990.
[13] A. Radulescu and A. J. C. Van Gemund. Fast and effective task scheduling in heterogeneous systems. In HCW '00: Proceedings of the 9th Heterogeneous Computing Workshop, page 229, Washington, DC, USA, 2000. IEEE Computer Society.
[14] T. L. Adam, K. M. Chandy, and J. R. Dickson. A comparison of list schedules for parallel processing systems. Commun. ACM, 17(12):685–690, 1974
[15] J.-J. Hwang, Y.-C. Chow, F. D. Anger, and C.-Y. Lee. Scheduling precedence graphs in systems with interprocessor communication times. SIAM J. Comput.,18(2):244–257, 1989.
[16] B. Kruatrachue and T. Lewis. Grain size determination for parallel processing. IEEE Softw., 5(1):23–32, 1988.
[17] M. Y. Wu and D. D. Gajski. Hypertool: A programming aid for message-passing systems. IEEE Trans. Parallel Distrib. Syst., 1(3):330–343, 1990.
[18] Runtong Zhang, Yannis A. Phillis and Vassilis S. Kouikoglou Fuzzy Control of Queuing Systems. © Springer-Verlag London Limited 2005.
[19] P. Brucker. Scheduling Algorithms. SpringerVerlag, 2004.
[20] J. Blythe, S. Jain, E. Deelman, Y. Gil, K. Vahi, A. Mandal, and K. Kennedy. Task scheduling strategies for workflow-based applications in grids. In CCGRID,pages 759–767,2005.

زیرنویس ها

[1] *Task*
[2] *Job*
[3] *Directed Acyclic Graph*
[4] *Polynomial time*
[5] *Heuristic*
[6] *List scheduling*
[7] *Clustering*
[8] *Task duplication*
[9] *Guided random search*
[10] *Static*
[11] *Dynamic*
[12] *Cost function*
[13] *Homogenous*
[14] *Heteregenous*
[15] *Scheduling with Fuzzy Weight assigning*
[16] *Application*
[17] *Task Graph*
[18] *Entry node*
[19] *Exit node*
[20] *Modified Critical Path*
[21] *Dynamic Level Scheduling*
[22] *Dynamic Level*
[23] *Dynamic Path*
[24] *Dynamic Critical Path*
[25] *Heterogeneous Earliest Finish Time*
[26] *Scheduling Different Capabilities*
[27] *Mapping Heuristic*
[28] *Fast Critical Path*
[29] *Insertion Scheduling Heuristic*
[30] *Fuzzification*
[31] *Inference engine*
[32] *Defuzzification*
[33] *rank*
[34] *Ready time*
[35] *Finish time*
[36] *Tardiness*
Total flow time [37]
[38] *Pseudo code*
[39] *Comparison metrics*
[40] *Normalized Scheduling Length*
[41] *Capable processors*
[42] *Average Percentage Degradation*
[43] *Communication to Computation Ratio*