گروه بندی دو مرحله ای کارها با استفاده از الگوریتم های ژنتیکی و اِعمال سیاست های مختلف زمانبندی در گریدهای محاسباتی

صادق وهابزاده زرگری [*]، عادل ترکمان رحمانی [†]

چکیده

گریدهای محاسباتی با توان عملیاتی زیاد، به طور گسترده برای اجرای کاربردهای محاسباتی تشکیل شده از کارهای مرتبط به هم مورد استفاده قرار می گیرند. پردازش این کاربردها به صورت رویهم رفته در گرید، هزینه و زمان زیادی را در بر دارد که ناشی شده از سربار به وجود آمده از (۱) ارسال کارها به منابع، (۲) پردازش کارها و (۳) انتقال نتیجه پردازش شده به سمت کاربر می باشد.

در این مقاله استراتژی زمانبندی مبتنی بر گروه‌بندی دومرحله ای، با اِعمال الگوریتم‌های ژنتیکی توسعه یافته‌است. الگوریتم-های ژنتیکی ابتدا در مرحله اول، یک کاربرد متشکل از کارهای وابسته به هم را به مجموعه ای از کاربردهای وابسته به هم تبدیل می کند و سپس در مرحله دوم با استفاده از اِعمال روش گروه‌بندی ژنتیکی بر روی تک‌تک کاربردهای حاصل، آنها را زمانبندی می کند. در ضمن سعی شده است امکان اِعمال سیاست‌های مختلف زمانبندی بر اساس کمینه نمودن هزینه و زمان اجرا وجود داشته باشد. نتایج و جزئیات ارزیابی با استفاده از اجرای یک شبیه ساز مبتنی بر شبیه ساز *GidSim* نشان داده شده است.

کلمات کلیدی

وابستگی کارها، گروه بندی دو مرحله ای کار ها، زمانبندی در سطح کاربرد، گریدهای محاسباتی با توان عملیاتی زیاد، کاربردهای متشکل از وظیفه های مجزا، الگوریتم های ژنتیکی

Two-stage Grouping Algorithm using Genetic Algorithms Based on Different Scheduling Policies on Computational Grids

Sadegh Vahabzadeh Zargari[‡], Adel Torkaman Rahmani[§]

Abstract

Although high throughput computational Grids have been used extensively for executing applications with compute-intensive jobs, there exist several applications with a large number of lightweight jobs. The overall processing undertaking of these applications involves high overhead time and cost in terms of (i) job transmission to and from Grid resources and, (ii) job processing at the Grid resources.

In this paper, we present two-stages scheduling strategy and use genetic algorithms to optimizing both of those stages. In the first stage a collaborative-tasks application transforms to subset of bag-of-tasks applications. In the second stage, we apply job grouping algorithm to scheduling these applications. Our scheduling strategy uses both of cost-based and time-based policies that use genetic algorithms for performing dynamic job grouping activity at runtime. The detailed analysis is conveyed by running simulations.

Keywords

Job Dependencies, 2-stage Job Grouping, Application Level Scheduling, High Throughput Computational Grids, Bag-of-Tasks Applications, Genetic Algorithms

[*] دانشگاه علم و صنعت ایران، دانشکده مهندسی کامپیوتر، Vahabzadeh@comp.iust.ac.ir

[†] عضو هیأت علمی دانشگاه، دانشگاه علم و صنعت ایران، دانشکده مهندسی کامپیوتر، Rahmani@iust.ac.ir

[‡] Iran University of Science and Technology, Computer Engineering Department, Vahabzadeh@comp.iust.ac.ir

[§] Academic Staff, Iran University of Science and Technology, Computer Engineering Department, Rahmani@iust.ac.ir

۱- مقدمه

با پدید آمدن ایده گریدهای محاسباتی، که توسط فوستر و کِسِلمن [۱] در سال ۱۹۹۹ مطرح گردید، یک سکوی نرم افزاری جدید برای اجرای کاربردها بر روی تعداد زیادی از منابع محاسباتی نامتجانس و توزیع شده، و با عبور از حوزه های مدیریتی، و اِعمال سیاست های مختلف به وجود آمد. با توجه به تعریف، یک کاربرد مجموعه ای از وظیفه ها و کارهای متعدد است و هر کار[۱] به عنوان یک واحد اتمی از محاسبات کاربرد تعریف می گردد.

تمام کاربردها از پتانسیل لازم برای مورد اجرا واقع شدن در گرید محاسباتی برخوردار نیستند و امکان دارد هزینه و زمان اجرای برخی از کاربردها در گرید بسیار بالاتر از اجرای آن کاربرد در یک ماشین خاص باشد. با وجود این، کلاس ایده آلی از کاربردها وجود دارد که از پتانسیل بسیار بالایی برای اجرا شدن در گرید برخوردار هستند. این کاربردها از یک سری وظیفه های مستقل و مجزا[۲] که نیازی به ارتباط درون پردازشی ندارند، تشکیل شده اند[۲]. هر یک از کارها و وظیفه های موجود در این کاربردها را می توان بدون توجه به سایر کارها زمانبندی و اجرا نمود. اگر این نوع کاربردها را در رده اول قرار دهیم، کاربردهای رده پایین تری هم وجود دارند. به عنوان مثال می توان به کاربردهایی اشاره نمود که از مجموعه ای از وظیفه های مرتبط و کارهای وابسته به هم تشکیل شده اند[۶]. در این کاربردها به دلیل وابستگی های موجود، پتانسیل بالای رده اول برای مورد اجرا واقع شدن در گرید وجود ندارد.

یک چالش مهم تبدیل کاربردهای رده پایین تر به کاربردهای رده اول که از قابلیت موازی سازی بالاتری برای مورد اجرا واقع شدن در گرید برخوردار هستند، می باشد.

چالش دیگر در این زمینه می تواند اجرای کاربردهای رده اول در یک محیط گرید محاسباتی باشد. با توجه به اینکه اجرای هر کاربرد عبارت است از زمانبندی کارهای تشکیل دهنده آن کاربرد، چالش مهم دیگر و قابل بحث در این زمینه، زمانبندی این کارها می باشد.

با توجه به تعریفی که توسط بِرمن، فاکس و هِی [۴] ارئه شده است، یک زمانبند[۳] مسئول انتخاب مناسبترین ماشین ها و منابع محاسباتی برای پردازش کارها در گرید می باشد و این انتخاب با هدف رسیدن به توان عملیاتی بالاتر، و با توجه به مشخصه های هر کاربرد ، دسترس پذیری، نیازمندیها و قابلیت پردازشی منابع انجام می پذیرد. مثلا در مورد کاربردی که از تعداد زیادی کار با مشخصه پردازشی کم وزن[۴] تشکیل شده است، کل زمان ارتباط میان هر کار و منابع بسیار بالاتر از کل زمان محاسبه و پردازش آن کار می باشد. راه حل مناسب برای رفع این مشکل، گروه بندی و جمع کردن کارهای ریزدانه[۵] در کنار هم و ایجاد کارهای درشت دانه[۶] یا مِتا کار[۷]، به منظور کاهش هزینه ارتباط و زمان پاسخ است[۵]. از محاسن این روش، واگذاری پروسس زمانبندی به برنامه و کاربرد سمت کاربر، و از معایب آن لزوم آگاهی محیط توسعه کاربرد از پیچیدگیهای موجود در گرید است. با استناد به انتشارات آی بی اِم در مورد قابل اجرا کردن کاربردها در گرید[۱۲و۶]، گنجاندن بسیاری از پیچیدگیهای موجود گرید و امکانات مدیریت منابع در خود کاربرد، در جهت دادن قابلیت گریدی به کاربرد انجام می شود و کاربرد می تواند به صورت پویا با موجودیت های مختلف گرید ارتباط برقرار نماید.

تا کنون روشهای گوناگونی برای انجام زمانبندی ارائه شده است. در این مقاله با توجه به روش زمانبندی اکتشافی [۳] و روش گروه‌بندی کارها [۱۳و۵] که برای کاربردهای متشکل از وظیفه‌های مجزا ارائه شده اند، و با احترام به روش زمانبندی گروه‌بندی دو مرحله‌ای کارها [۱۴] برای کاربردهای متشکل از وظیفه های مرتبط، روش زمانبندی بهتری با استفاده از الگوریتم‌های ژنتیکی ارائه شده است.

ادامه این مقاله از بخش های زیر تشکیل شده است. در بخش دوم پیشینه کار معرفی شده، و کارهای انجام شده قبلی تحت بررسی قرار گرفته است. در بخش سوم روش گروه بندی دومرحله‌ای ارائه شده است. در بخش چهارم الگوریتم پیشنهادی به همراه جزئیات محیط شبیه سازی بیان شده است. بخش پنجم به تحلیل و بررسی نتایج شبیه سازی اختصاص یافته است، و در بخش ششم ضمن نتیجه گیری نهایی ، برنامه ها و کارهای آتی در زمینه مقاله ارائه شده، بیان شده است.

۲- کارهای مرتبط انجام شده

پروژه GrADSolve[۸] [۹] یک سیستم RPC[۹] را به منظور انجام پردازش ها و محاسبات موازی در گرید، ارائه می دهد. زمانبندی GrADSolve، بر اساس زمانبندی در سطح کاربرد و پروژه AppLeS[۱۰] می باشد و از ترکیب دو پروژه عظیم مرتبط با گرید به نام های GrADS[۱۱] و NetSolve به وجود آمده است. GrADS فن آوریها و ابزارهای لازم را برای توسعه و اجرای کاربردها در محیط گرید فراهم می آورد و یک روش زمانبندی در سطح کاربرد، ارائه می دهد. NetSolve یک سیستم گرید محاسباتی بر اساس RPC است که با

---

[1]- Job
[2]- Bag-of-tasks applications
[3]- Scheduler
[4]- Light weight jobs
[5]- Fine-grained jobs
[6]- Coarse-grained jobs
[7]- Meta-jobs
[8]- Grid Application Development Solver
[9]- Remote Procedure Call
[10]- Application Level Scheduling
[11]- Grid Application Development Software

هدف اجرای کاربردهای موازی از راه دور به وجود آمده است. هسته اصلی پروژه GrADSolve بر اساس بانک اطلاعات XML[12] است.

در روش زمانبندی اکتشافی[13] [۳] با ارائه الگوریتم Qsufferage، سعی شده است یک روش زمانبندی با احترام به پارامترهای کیفیت سرویس و برای رده خاصی از کاربردها که متشکل از وظیفه های مجزا هستند، ارائه گردد. هدف این روش کمینه کردن زمان پاسخ می باشد. نسبت پاسخ[14] (rr) به عنوان پارامتر کیفیت سرویس (QoS) به صورت رابطه ۱ تعریف شده است.

$$rr = \frac{the \therefore duration \therefore of \therefore running}{the \therefore completed \therefore time - the \therefore submited \therefore time} \qquad (۱)$$

در روش زمانبندی مجزا[15] [۱۰] زمانبند و قطعات ویژه کاربرد که در انجام پروسس زمانبندی مورد نیاز هستند، از هم تفکیک شده اند. زمانبند دارای یک پروسیجر جستجو است که اطلاعات ورودی آن سیستمهای اطلاعاتی گرید، مدل کارآیی[16] ، داده های تعیین احتمالات، لیست تطبیق[17]، و اجزای دیگر هستند و خروجی آن زمانبندی نهایی می باشد.

مدل کارآیی هر کاربرد با استفاده از بررسی آن کاربرد، و با هدف پیش بینی زمان اجرا و میزان حافظه مورد نیاز به دست می آید. در این روش سعی شده است، با توجه به ماهیت پویا بودن گرید، درصد خطای پیش بینی[18] کاهش یابد. درصد خطای پیش بینی با فرمول رابطه ۲ محاسبه می شود.

$$predictionError = 100 * \frac{predictionTime - actualTime}{actualTime} \qquad (۲)$$

در روش زمانبندی تطبیقی[19] [۷] با ارائه مدلهای پیش بینی، زمانبندی کارها در هر دو سطح، یعنی هم در سطح سیستم و هم در سطح کاربرد انجام می شود. در سطح کاربرد، با اِعمال الگوریتم های ژنتیکی، ضمن تخصیص بهینه کارها به منابع، زمان متوسط تکمیل کارها کاهش می یابد. در این روش، گونه های مختلف جمعیت ژنتیکی با کنار هم قرار دادن تعداد کارهای احتمالی تخصیص یافته به هر منبع به دست می آید. طول هر عضو جمعیت برابر تعداد منابع و هر ژن نشان دهنده تعداد کارهای تخصیص یافته به یک منبع است. به عنوان مثال با داشتن n کار و m منبع اعضای جمعیت ژنتیکی به صورت n_1 n_2 n_3 . . . n_m خواهد بود و شرط صحت، برابر بودن مجموع تعداد کارهای منابع با تعداد کل کارها طبق رابطه ۳ می باشد.

$$n = \sum_{i=1}^{m} n_i \qquad (۳)$$

در روش زمانبندی Sub optimal [۸] با ارائه یک زمانبند فوقانی[20]، یک زمانبندی دو سطحی ارائه شده است. در سطح اول گره های محاسباتی قرار دارند که هر کدام زمانبند محلی و سیاست زمانبندی خاص خودشان را دارند و در سطح دوم زمانبند فوقانی قرار دارد که هدف آن پیدا کردن دنباله ای مناسب از کارها برای تخصیص به هر گره با استفاده از اِعمال الگوریتم های ژنتیکی می باشد.

در روش زمانبندی با استفاده از گروه بندی کارها[۵]، یک روش زمانبندی کارها برای کاربردهایی که از وظیفه های ریزدانه و مجزا تشکیل شده اند، ارائه شده است. گروه بندی کارها با هدف تبدیل چندین کار ریزدانه به یک کار درشت دانه و به منظور کاهش نرخ ارسال کارها انجام می گیرد. گروه بندی بر اساس پارامتر ویژه ای به نام Granularity Size انجام می شود. Granularity Size مدت زمانی است که انتظار می رود یک کار در یک منبع محاسباتی مورد پردازش قرار گیرد. این پارامتر در قابلیت پردازشی منابع ضرب می شود تا تعداد مورد انتظار کارهای قابل پردازش در یک زمان معین، برای هر منبع محاسباتی به دست آید. در روش گروه‌بندی با استفاده از الگورینم‌های ژنتیکی[۱۳] ضمن بهینه‌سازی روش گروه‌بندی، انجام زمانبندی با توجه به سیاست‌های مختلف میسر شده است. روش زمانبندی مبتنی بر گروه‌بندی دو مرحله‌ای کارها[۱۴] الگوریتم زمانبندی را برای کاربردهای متشکل از وظیفه های وابسته به هم قابل استفاده نموده است.

۳- روش گروه بندی دو مرحله‌ای

با وجود اینکه روش زمانبندی تطبیقی با ارائه مدلهای پیش بینی در هر دو سطح سیستم و کاربرد، زمانبندی مناسبی ارائه نموده است، اما به ماهیت واقعی بسیاری از موجودیت ها توجه نداشته است. به عنوان مثال فقط به تعداد کارها توجه شده است و بار پردازشی کارها مورد بحث قرار نگرفته است. در ضمن به هزینه اجرای روش توجه نشده است.

روش گروه بندی کارها با آنکه توانسته است به میزان قابل اعتنایی هزینه و زمان پردازش کارها را کاهش دهد، اما فقط برای زمانبندی کاربردهای رده اول که از کارهای مجزا تشکیل شده اند، ارائه شده است. این روش کارآیی لازم را در مورد کاربردهایی که از وظیفه های مرتبط و کارهای وابسته به هم تشکیل شده اند، ندارد.

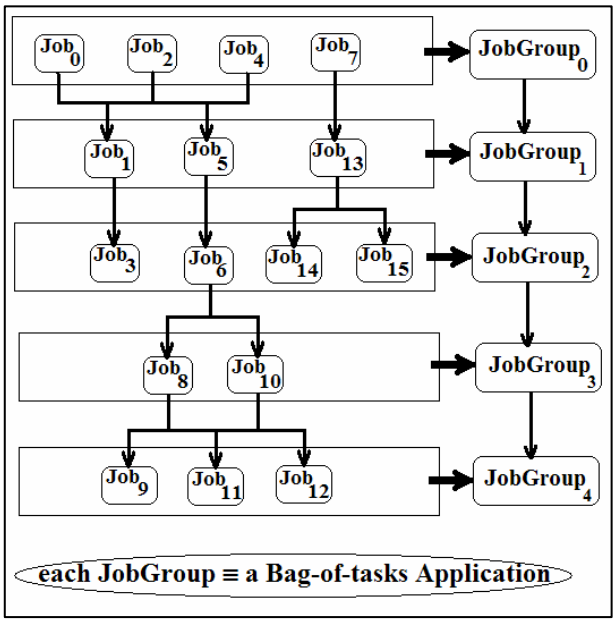
روش پیشنهادی با اِعمال الگوریتم های ژنتیکی، یک استراتژی گروه بندی دو مرحله ای را برای زمانبندی کارها ارائه می دهد. این روش در مرحله اول یک کاربرد متشکل از کارهای وابسته به هم را به مجموعه ای از کاربردهای وابسته به هم که از رده اول هستند، تبدیل می کند و سپس در مرحله دوم با استفاده از اِعمال روش گروه بندی[۱۳] بر

12- eXtensible Markup Language
13- Heuristic Scheduling
14- Response Ratio
15- Decoupled Scheduling
16- Performance Model
17- Match List
18- Prediction Error
19- Adaptive Scheduling
20- Superscheduler

روی تک تک کاربردهای حاصل، آنها را به ترتیب وابستگی، زمانبندی می کند.

بنابراین لازم است که ابتدا کاربردهای متشکل از کارهای وابسته به هم، به کاربردهای رده اول تبدیل شوند. برای این منظور باید کارها را به صورتی گروه بندی نمود که به جای کارهای وابسته به هم، گروه های کاری وابسته به هم داشته باشیم و کارهای دسته بندی شده در هر گروه کاری نیز مستقل از هم باشند. در این صورت می توان با هر گروه کاری حاصل شده، مثل یک کاربرد رده اولی برخورد نمود. استراتژی انجام این کار در شکل ۱ با ذکر مثالی بیان شده است.

همانطور که در شکل ۱ قابل رؤیت است، کارها با توجه به سطح وابستگی که در آن قرار گرفته‌اند گروه بندی می شوند.



شکل (۱): یک مثال از استراتژی مرحله اول گروه بندی

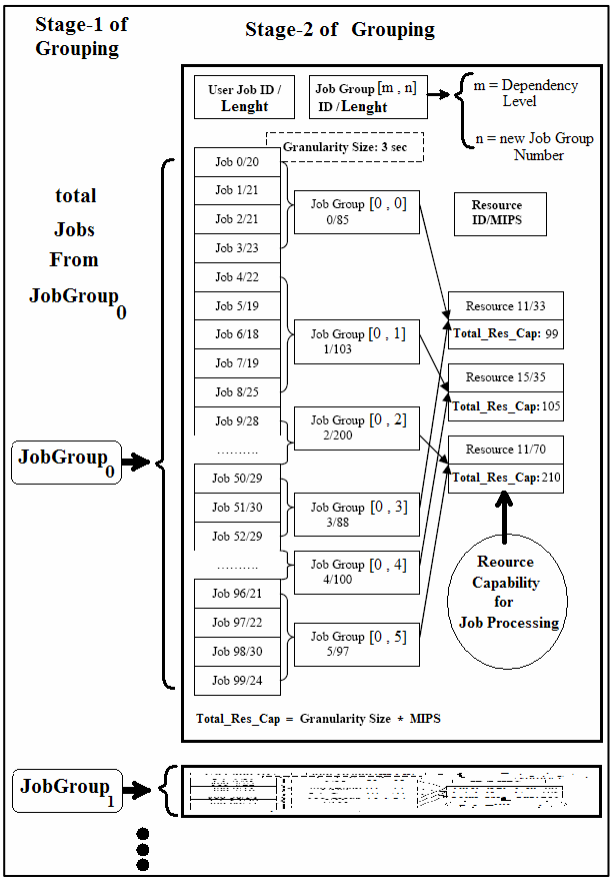
هر یک از گروه های کاری به دست آمده از مرحله اول گروه بندی را می توان هم ارز یک کاربرد رده اول در نظر گرفت. بنابراین در مرحله دوم می توان بر روی هر یک از گروه های کاری به دست آمده از مرحله اول، الگوریتم گروه بندی را که برای کاربردهای رده اول ارائه شده است اِعمال نمود.

آنچه که مسلم است تعداد دفعات اِعمال الگوریتم گروه بندی در مرحله دوم برابر تعداد گروه های کاری ایجاد شده است و از آنجا که تعداد گروه های کاری ایجاد شده با تعداد سطوح وابستگی نسبت مستقیم دارد، پارامتر Dependency Level اهمیت بالاتری پیدا می کند

در شکل ۲ استراتژی مرحله دوم گروه بندی کارها با استفاده از ذکر یک مثال نشان داده شده است. همانطور که از شکل ۲ قابل فهم است، الگوریتم گروه بندی مرحله دوم برای هر یک از گروه های کاری به دست آمده از مرحله اول یک بار اجرا می شود.

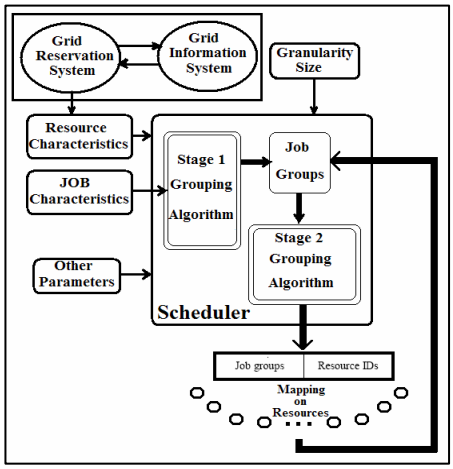
گروه بندی کارها با هدف تبدیل چندین کار ریزدانه به یک کار درشت دانه و به منظور کاهش نرخ ارسال کارها انجام می گیرد. گروه بندی بر اساس پارامتر ویژه ای به نام Granularity Size انجام می شود.

Granularity Size مدت زمانی است که انتظار می رود یک کار در یک منبع محاسباتی مورد پردازش قرار گیرد. این پارامتر در قابلیت پردازشی منابع ضرب می شود تا تعداد مورد انتظار کارهای قابل پردازش در یک زمان معین، برای هر منبع محاسباتی به دست آید.



شکل (۲): یک مثال از استراتژی مرحله دوم گروه بندی

شمای کلی روش گروه بندی دو مرحله ای در شکل ۳ نشان داده شده است. به علت پویا بودن گرید، برای اطمینان از در دسترس بودن منابع در زمان اجرای کارها، وجود یک سیستم رِزِرو منابع که با یک سیستم اطلاعاتی در ارتباط باشد، ضروری است.



شکل (۳): شمای کلی زمانبند ارائه شده

اطلاعات منابع، کارها، و پارامتر Granularity Size، و سایر پارامترهای لازم به زمانبند اِعمال می شوند و زمانبند، در مرحله اول با گروه بندی کارها بر اساس وابستگی آنها، کارهای وابسته به هم را به گروه های کاری وابسته به هم تبدیل می کند و در مرحله دوم با استفاده از روش گروه بندی، کارهای موجود در هر گروه کاری را زمانبندی می کند و ماتریس نگاشت کارها را به منابع مناسب تولید می کند. هر گروه کاری منتظر دریاقت نتیجه پردازش شده گروه کاری قبلی می ماند و با استفاده از روش گروه بندی زمانبندی می شود.

شبه کد الگوریتم مرحله اول گروه بندی در خطوط ۱ الی ۲۳ شکل ۴ ارائه شده است و با توجه به مثال ذکر شده در شکل ۱ نیاز به توضیحات بیشتر وجود ندارد. در خط ۱۱ گروه کاری جدید مربوط به یک سطح وابستگی ایجاد می شود و در خط ۱۶ وابستگی ها بررسی می شود. از نتایج مهم این الگوریتم معرفی پارامتر سطح وابستگی یا Dependency Level است و هسته مرکزی آن، افزودن کارها به گروه کاری سطح وابستگی مربوطه است که در خط ۱۸ بیان شده است.

```
// Stage 1 Grouping Algorithm
1 ResourceList := Resource_characteristics(GRS,GIS );
2 JobList := Job_characteristics_from_application();
3 DependencyLevel := 0;
4 for i:= 0 to JobList.Size-1 do
5   JobList_i.used=false;
6 endfor
7 for i:= 0 to JobList.Size-1 do
8  if JobList_i.used = true then
9   continue;
10  endif
11  JobGroup.AddnewGroup ( DependencyLevel );
12  for j:= 0 to JobList.Size-1 do
13   if JobList_j.used = true  then
14    continue;
15   endif
16   if  ( JobList_j.beforJob = Null )  or
        ( JobList_j.beforJob.used = true ) then
17    JobList_j.used = true;
18    JobGroup_dependencyLevel.add( JobList_j )
19   endif
20  endfor
21  DependencyLevel ++;
22 endfor
23  DependencyLevel --;
```

شکل (۴): شبه کد مرحله اول گروه بندی

همچنین شبه کد الگوریتم مرحله دوم گروه بندی در خطوط ۲۴ الی ۵۵ شکل ۵ ارائه شده است. در این مورد نیز با توجه به مثال ذکر شده در شکل ۲ نیاز به توضیحات اضافی احساس نمی شود. در خط ۲۵، حلقه تکرار الگوریتم مرحله دوم به تعداد سطوح وابستگی قرار دارد. در خط ۲۶ کارهای موجود در گروه کاری ایجاد شده از مرحله اول گروه بندی که مربوط به سطح وابستگی جاری هستند، در لیست کارهای الگوریتم گروه بندی مرحله دوم قرار می گیرند.

در خطوط ۲۷ الی ۴۸ گروه بندی کارها با توجه به پارامتر Granularity Size انجام می گیرد. در خطوط ۴۹ الی ۵۱ گروه های کاری ایجاد شده به منابع ارسال می شوند و در خطوط ۵۲ الی ۵۴ ، الگوریتم در انتظار دریافت نتایج پردازش شده در منابع می ماند.

در خط ۵۵ شاهد ادامه تکرار حلقه for و تکرار دوباره خطوط ۲۶ الی ۵۴ برای کارهای موجود در سطوح وابستگی پایین تر که منتظر دریافت نتیجه پردازش کارها در سطوح بالاتر بودند، خواهیم بود.

```
// Stage 2 Grouping Algorithm
24 NextResource:= 0;
// Dependency Loop
25 for deLevel := 0 to DependencyLevel-1 do
// get Jobs List from current Dependency Level
26  JobList := JobGroups_dependencyLevel ;
// Grouping Algorithm
27  m := 0;
28  for i:= 0 to JobList.Size-1 do
29   for j:= NextResource to ResourceList.Size-1 do
30    if  NextResource = ResourceList.Size-1 then
31     NextResource := 0;
32    endif
33    Total_JobLoad := 0;
34    Total_ResourceCapability :=
          ResourceList_j.MIPS * Granularity_Size;
35    while(Total_JobLoad ≤ Total_ResourceCapability
           and  i ≤  JobList.Size-1 ) do
36     Total_JobLoad :=
          Total_JobLoad  +  JobList_i.JobLenght;
37     i++;
38    endwhile
39    i--;
40    if Total_JobLoad > Total_ResourceCapability then
41     Total_JobLoad :=
          Total_JobLoad  -  JobList_i.JobLenght;
42     i--;
43    endif
44    GroupJobList_m :=
      Add_new_Job (Total_JobLoad , assign unique ID);
45   GroupJobList_m.TargetResource:=R esourceList_j.ID;
46    m++;
47  endfor
48 endfor
// Sending jobs for computation
49 for i:= 0 to GroupJobList.Size-1 do
50  Send GroupJobList_i to TargetResource
     for  job  computation;
51 endfor
// Receiving computed results
52 for i:= 0 to GroupJobList.Size-1 do
53  Receive computed GroupJobList_i
    from TargetResource;
54 endfor
55 endfor
```

شکل (۵): شبه کد مرحله دوم گروه بندی

زمان کل پردازش با استفاده از رابطه (۴) محاسبه می شود. همانطور که قابل مشاهده است، زمان سربار حاصل از عمل گروه بندی هم به کل زمان افزوده شده است.

$$\begin{aligned} &\textit{Total Processing Time} = \\ &\quad \textbf{\textit{Grouping Algorithm Time}} + \\ &\quad \textbf{\textit{Genetic Algorithm Time}} + \\ &\quad \textit{Total Processing Overhead Time} + \\ &\quad \textit{Total JobGroups Computing Time} + \\ &\quad \textit{Time taken to receive all computed jobs} \end{aligned} \qquad (۴)$$

۴- روش پیشنهادی ومحیط شبیه سازی

در روش پیشنهادی، الگوریتم های ژنتیکی یک بار در مرحله اول گروه بندی و بار دیگر در مرحه دوم آن اِعمال می گردد.

در مرحله اول هر عضو از جمعیت[21] ژنتیکی، از جایگشت های تصادفی گروه های کاری حاصل از عمل گروه‌بندی به وجود می آید.

به عنوان مثال اولین عضو در جمعیت ژنتیکی حاصل، به صورت رابطه ۵ خواهد بود.

$$\textbf{Population}_1 = \textbf{JobGroup}_1\ \textbf{JobGroup}_2 \ldots \textbf{JobGroup}_{\textbf{dependencyLevel}} \qquad (۵)$$

[21] Population

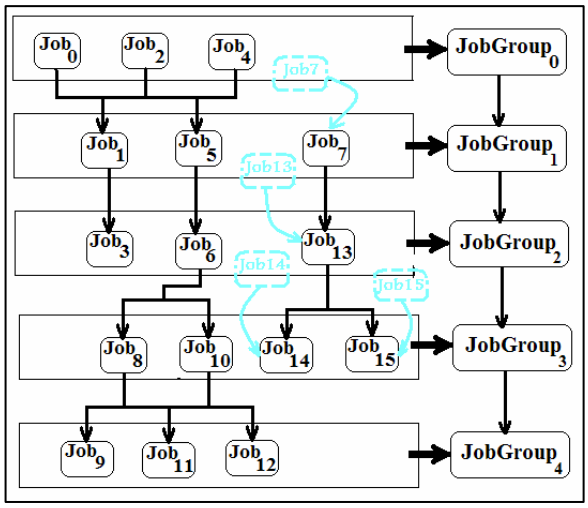
تابع برازندگی[۲۲] هم به گونه ای انتخاب می‌شود که حداکثر تعادل بار[۲۳] را در بین گروههای کاری مختلف داشته باشیم. بین جمعیت های ایجاد شده، جمعیت هایی مورد قبول خواهند بود که وابستگی کارها را همانند وابستگی موجود بین آن کارها در جمعیت اولیه حفظ کنند. برای این کار ابتدا تابع Δ_{ij} را به صورت رابطه ۶ تعریف می‌کنیم.

$$\Delta ij = \text{JobGroup}_i.\text{NumOfJobs} - \text{JobGroup}_j.\text{NumOfJobs} \quad (۶)$$

Δ_{ij} اختلاف تعداد کارهای موجود در گروه کاری i و گروه کاری j را نشان می‌دهد و تابع برازندگی به صورت رابطه ۷ خواهد بود.

$$\text{Fitness Function} = \sqrt{\sum_{(For\_Each\_i,j)} (\Delta_{ij})^2} \quad (۷)$$

در شکل ۶ می‌توانید نتیجه اجرای الگوریتم ژنتیک را بر روی مثال شکل ۱ و مرحله اول گروه بندی ببینید. همانطور که قابل مشاهده است اجرای الگوریتم ژنتیک منجر به ایجاد یک تعادل باری میان تعداد کارهای موجود در گروههای کاری شده است و در هر گروه کاری به طور متوسط ۳ کار قرار گرفته است و فقط JobGroup_3 دارای ۴ کار است. یعنی ۱۶ کار موجود به طور مساوی در بین گروه‌های کاری تقسیم شده‌اند.



شکل (۶): نتیجه اجرای الگوریتم ژنتیک در مرحله اول

در مرحله دوم هر عضو از جمعیت ژنتیکی، از جایگشت های تصادفی کارها به وجود می‌آید و با استفاده از یک تابع برازندگی مناسب، جمعیت نخبگان انتخاب می شود. به عنوان مثال اگر n کار داشته باشیم اولین عضو از جمعیت ژنتیکی به صورت رابطه ۸ خواهد بود.

$$\text{Population}_1.\text{JobList} = \text{Job}_1\ \text{Job}_2\ \text{Job}_3\ \ldots\ \text{Job}_n \quad (۸)$$

اگر سیاست زمانبندی، کمینه نمودن زمان[۲۴] محاسبه باشد، تابع برازندگی از رابطه ۹ محاسبه می گردد.

$$Fitness\_function = \frac{|Total\_JobLoad - Total\_ResourceCapacity|}{Total\_JobLoad} \quad (۹)$$

در صورتیکه سیاست زمانبندی، کمینه نمودن هزینه[۲۵] محاسبه باشد، تابع برازندگی از رابطه ۱۰ محاسبه می گردد.

$$Fitness\_function = ResourceList_j.Cost * \frac{Total\_JobLoad}{Total\_ResourceCapacity} \quad (۱۰)$$

اگر سیاست زمانبندی این باشد که یک مُوازنه در کمینه نمودن هزینه محاسبه و زمان انجام آن برقرار شود، تابع برازندگی از رابطه ۱۱ محاسبه می گردد.

$$Fitness\_function = ResourceList.Cost * \frac{|Total\_JobLoad - Total\_ResourceCapacity|}{Total\_ResourceCapacity} \quad (۱۱)$$

در هر تکرار با اِعمال توابع جهش[۲۶] ژنتیکی و پیوند[۲۷] ژنتیکی، اعضای جمعیت به جواب بهینه همگرا می شوند. نحوه تنظیم پارامترهای ژنتیکی، تاثیر قابل توجهی در سرعت همگرایی و نتیجه نهایی دارد. بنابراین پیدا کردن یک مُوازنه[۲۸] مناسب در تنظیم پارامترهای ژنتیکی از چالش های مهم می باشد.

جهت مقایسه الگوریتم پیشنهادی با روش های مشابه، یک محیط شبیه سازی مبتنی بر شبیه ساز GridSim [۱۱] که به زبان جاوا نوشته شده، پیاده سازی شده است.

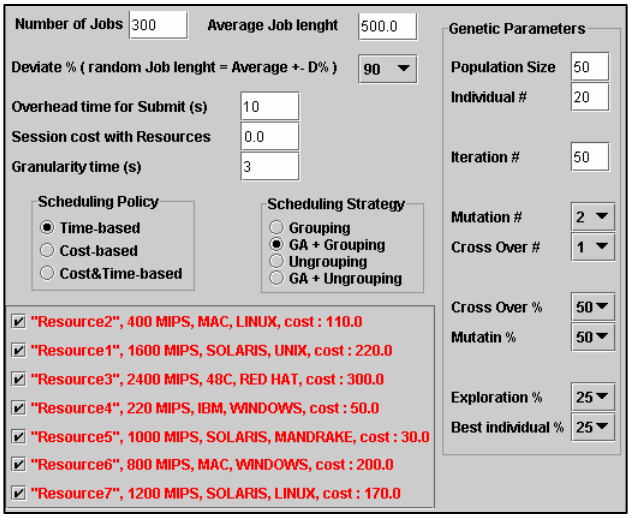
GridSim یک جعبه ابزار است که برای مدل کردن و شبیه سازی مدیریت منابع گرید و زمانبندی اجرای کارها طراحی شده است. این جعبه ابزار متشکل از دو بسته نرم افزاری به نامهای gridsim و simjava می باشد.

کلاسهای موجود در بسته gridsim شبیه سازی موجودیت های گرید، هستند. به عنوان مثال، کاربر گرید با کلاس GridUser ، کارها با کلاس Gridlet ، و منابع با کلاس GridResource نمونه سازی می شوند.GridSim از امکانات لازم جهت ایجاد کارهای با طول های متفاوت و استاندارد، تعیین توپولوژی، ارسال کارها به منابع، دریافت نتیجه محاسبه، و شبیه سازی زمان محاسبه برخوردار است.

با توجه به شکل ۷ و محیط شبیه سازی، اندازه متوسط کارها برابر ۵۰۰ در نظر گرفته شده است و برای تولید کارها با طولهای متفاوت، حداکثر میزان انحراف ۹۰٪ ± در نظر گرفته شده است. با این توضیح که نتیجه آزمون شبیه سازی با تعیین تعداد کل کارها برابر ۱۰۰ ، ۲۰۰ ، ۵۰۰، ۱۰۰۰ و ۵۰۰۰ مشابه هم بوده است، و با توجه به اینکه پارامتر Dependency Level نقش تعیین کننده تری دارد، تعداد کارها در

Fitness function-[22]
Load Balancing-[23]
Time-based-[24]
Cost-based-[25]
Mutation-[26]
Cross over-[27]
Trade-off-[28]

محیط شبیه سازی ثابت و برابر ۵۰۰ در نظر گرفته شده است. پارامتر Granularity Size برابر عدد ۳ و زمان سربار ارسال ودریافت کارها برابر ۱۰ ثانیه فرض شده است.



Name	Platform (Architecture , OS)	MIPS	Cost
R1	SOLARIS , Unix	1600	220
R2	MAC , Linux	400	110
R3	48C , Redhat	2400	300
R4	IBM , Windows	220	50
R5	SOLARIS , Mandrake	1000	30
R6	MAC , Windows	800	200
R7	SOLARIS , Linux	1200	170

شکل (۷): محیط شبیه سازی ومشخصات منابع

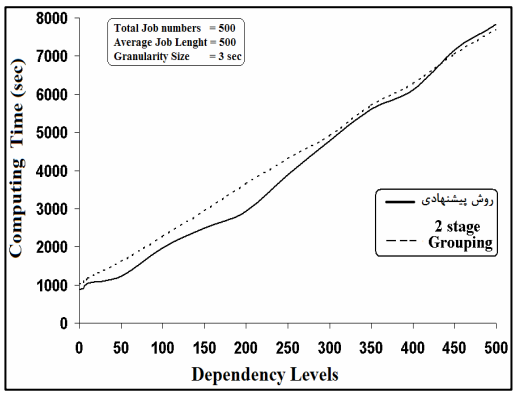
۵- نتایج شبیه سازی

نتایج به دست آمده حاکی از این است که روش پیشنهادی، هزینه و زمان اجرای خیلی بهتری نسبت به روش گروه‌بندی دومرحله ای دارد. در جدول شکل ۸ جزئی از نتایج حاصل شده از شبیه سازی نشان داده شده است.

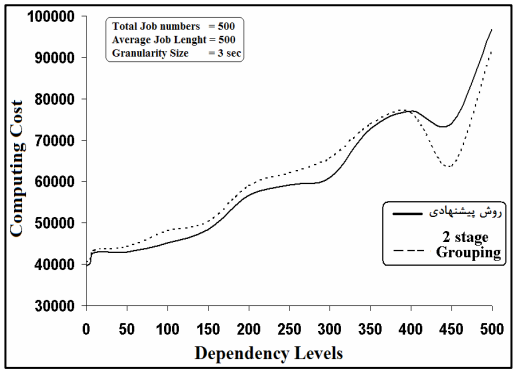
Dependency Levels	2-stage Grouping Algorithm			2-stage Grouping+Genetic Algorithm		
	Computing Cost	Computing Time	Groups#	Computing Cost	Computing Time	Groups#
0	40587	1015	84	39513	884.2	82
5	41243	1080	87	40368	912.3	84
10	43388	1182	93	42876	1042	89
50	44223	1616	114	42997	1223	107
100	48169	2280	154	45224	1972	141
150	50390	2951	197	48356	2491	178
200	59068	3656	240	56724	2936	235
250	62147	4315	284	59122	3892	249
300	65700	4917	320	61003	4781	304
350	73884	5704	372	72724	5605	362
400	76686	6286	411	76953	6118	409
450	63887	7057	461	74122	7156	459
499	91709	7702	500	96791	7824	500

شکل (۸): جزئی از نتایج حاصل از شبیه سازی

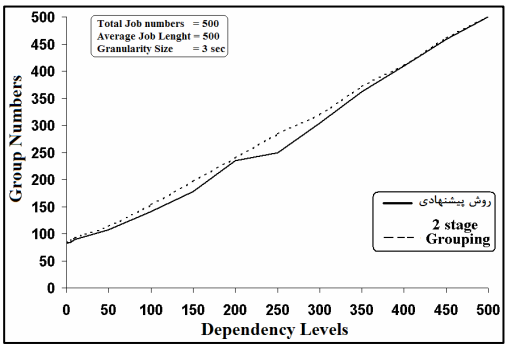
در شکل ۹ ، نتایج کل زمان پردازش و محاسبه کارها به زبان نمودار بیان شده است. با توجه به نمودار روش پیشنهادی تا قبل از رسیدن به سطح وابستگی حدود ۹۰٪ بهتر از روش دو مرحله ای می باشد.



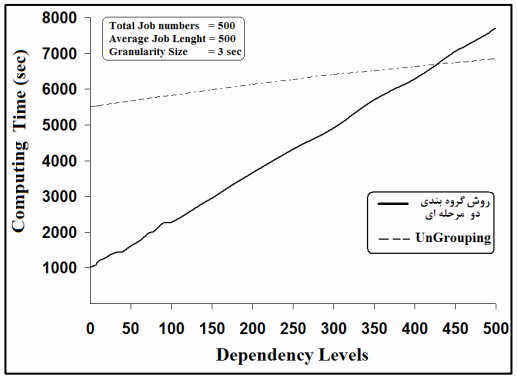
شکل (۹): مقایسه کل زمان محاسبه دو روش



شکل (۱۰): مقایسه هزینه محاسبه دو روش



شکل (۱۱): نمودار تعداد گروه ها



شکل (۱۲): مقایسه روش گروه‌بندی با روش Ungrouping

در نمودار شکل ۱۰ ، نتیجه ارزیابی هزینه محاسبه کارها به منعکس شده است. با توجه به نمودار، نمودار هزینه روش پیشنهادی در بیشتر موارد زیر نمودار هزینه روش گروه بندی دو مرحله ای و بهتر از آن می باشد. در ضمن با افزایش تعداد سطوح وابستگی، نمودار هزینه دو روش به سمت یکدیگر همگرا شده است که آن را به نحوی می توان تایید نتیجه حاصل از نمودار شکل ۹ دانست.

با توجه به نمودار شکل ۱۱ ، تعداد گروه ها در صورت اِعمال الگوریتم ژنتیک کاهش می یابد. در ضمن تعداد گروه های حاصل، متناسب با افزایش سطوح وابستگی و نزدیک شدن آن به حد نهایی که برابر تعداد کل کارها می باشد، افزایش می یابد و در نهایت برابر تعداد کل کارها می شود.

در شکل ۱۲ ، لزوم گروه‌بندی به زبان نمودار بیان شده است. با توجه به کارآیی بهتر اِعمال الگوریتم ژنتیک و با توجه به نمودار شکل ۱۲ می توان نتیجه گرفت که روش پیشنهادی نسبت به روش Ungrouping نیز از کارآیی به مراتب بهتری برخوردار می باشد.

۶- نتیجه گیری و کارهای آتی

با گسترش روزافزون گرید، مخصوصا گریدهای محاسباتی، مدیریت بهینه منابع و بهره وری از توان محاسباتی آنها از چالش های مهم و قابل بحث می باشد. با افزایش کاربردهای حجیم تشکیل شده از وظیفه های وابسته به هم و چالش اجرای آنها در گرید محاسباتی، زمانبندی صحیح و پویای این کاربردها، تاثیر قابل توجهی در مدیریت بهینه منابع دارد. در روش ارائه شده در زمانبندی تطبیقی، به ماهیت واقعی بسیاری از موجودیت ها توجه نشده است. روش گروه بندی کارها با آنکه توانسته است به میزان قابل اعتنایی هزینه و زمان پردازش کارها را کاهش دهد.

روش پیشنهادی، با اِعمال الگوریتم ژنتیک بر روی روش گروه بندی دو مرحله ای آن را تا حد قابل قبولی بهبود داده است. نتایج شبیه سازی حاکی از کارآیی بهتر روش پیشنهادی نسبت به روش گروه بندی دو مرحله ای دارد. با توجه به شکل ۹ و نتایج به دست آمده، ۱۰/۵ درصد بهبود در زمان محاسبه کارها حاصل شده است که این مقدار در مورد کاربردهایی که سطح وابستگی آنها زیر ۹۰٪ است به ۳۶/۸ درصد می رسد و از آنجاییکه معمولا سطح وابستگی کاربردها به ندرت به ۹۰٪ می رسد، بهبود به دست آمده قابل اعتنا می باشد. درضمن با توجه به شکل ۱۰ و نتایج حاصل از آن ۱/۴ درصد بهبود در هزینه محاسبه کارها حاصل شده است که این مقدار در مورد کاربردهایی که سطح وابستگی آنها زیر ۹۰٪ است به ۱۳/۳ درصد می رسد.

پیشنهاد می شود که در آینده بر روی زمانبندهای نامتمرکز و زمانبندهای مبتنی بر مدل های پیش بینی کارآیی، کارهای بیشتری صورت پذیرد.

مراجع

[1]. Foster, I. and Kesselman, C. (1999): *The Grid: Blueprint for a New Computing Infrastructure.* San Francisco, Morgan Kaufmann Publisher, Inc.

[2]. A.L. Rosenberg, *Optimal schedules for cycle-stealing in a network of workstations with a bag-of-tasks workload*, IEEE Trans. Parallel Distribut. Syst. 13 (2) (2002) 179–191.

[3]. Chuliang Weng, Xinda Lu (2003): *Heuristic scheduling for bag-of-tasks applications in combination with QoS in the computational grid*, Elsevier B.V.

[4]. Berman, F., Fox, G. and Hey, A. (2003): *Grid Computing– Making the Global Infrastructure a Reality.* London,Wiley.

[5]. Nithiapidary, M., Junyang, L., Nay, L.S., Srikumar, V., Anthony, S. and Rajkumar Buyya (2005): *A Dynamic Job Grouping-Based Scheduling for Deploying Applications with Fine-Grained Tasks on Global Grids,* GRIDS Laboratory-The University of Melbourne, Australia.

[6]. Bart, J., Luis, F., Norbert, B., Candice, G., Jean-Yves, G., Roman, S., Seong Yu (2003): *Enabling Applications for Grid Computing with Globus,*ibm.com/redbooks.

[7]. Yang Gao, Hongqiang Rong, Joshua Zhexue Huang (2004): *Adaptive grid job scheduling with genetic algorithms,* Elsevier B.V.

[8]. V. Di Martino *, M. Mililotti (2004): *Sub optimal scheduling in a grid using genetic algorithms,* Elsevier B.V.

[9]. Sathish S. Vadhiyara, and Jack J. Dongarra(2003): *GrADSolve-a grid-based RPC system for parallel computing with Application-Level Scheduling,* Elsevier Inc.

[10]. Holly Dail, Fran Berman, and Henri Casanova (2002): *A decoupled scheduling approach for Grid application development environments*, Elsevier Science (USA).

[11]. Buyya, R. and Murshed, M. (2002): *GridSim: A Toolkit for the Modeling, and Simulation of Distributed Resource Management, and Scheduling for Grid Computing,* Journal of Concurrency and Computation: Practice and Experience (CCPE), 14(13-15):1175-1220.

[12]. Arun, T., Michael, B., Fabiano, L., Huang, R., Linda, L., Paul, M., Jeff, M., Nasser, M., Karthik, S., Olegario, H., Luis, F. (2004): *Grid Services Programming and Application Enablement,* ibm.com/redbooks.

[۱۳]. **صادق وهابزاده زرگری، عادل ترکمان رحمانی (۲۰۰۶):** یک *روش زمانبندی پویا مبتنی بر گروه بندی کارها با استفاده از الگوریتم های ژنتیکی برای کاربردهای متشکل از کارهای ریزدانه در گرید های محاسباتی،* **یازدهمین کنفرانس انجمن کامپیوتر ایران (CSICC)، تهران.**

[۱۴]. **صادق وهابزاده زرگری، عادل ترکمان رحمانی (۲۰۰۶):** یک *روش زمانبندی پویا مبتنی بر گروه بندی دو مرحله ای کارها برای کاربردهای متشکل از کارهای وابسته به هم در گرید های محاسباتی،* **یازدهمین کنفرانس انجمن کامپیوتر ایران (CSICC)، تهران[1].**