دوره دوازدهم، شماره دوم، پاییز و زمستان ۱۳۹۸

مجله فناوری اطلاعات در طراحی مهندسی

Information Technology in Engineering Design

http://ited.sinaweb.net

زمان‌بندی وظایف پویا با استفاده از الگوریتم تبرید شبیه‌سازی‌شده هذلولی در محیط‌های پردازش موازی با منابع ناهمگن

علیرضا صادقی حصار [۱] سید رضا کامل * [۲] محبوبه هوشمند [۳]

(۱) گروه مهندسی کامپیوتر، واحد مشهد، دانشگاه آزاد اسلامی، مشهد، ایران.

(۲) گروه مهندسی کامپیوتر، واحد مشهد، دانشگاه آزاد اسلامی، مشهد، ایران. *

(۳) گروه مهندسی کامپیوتر، واحد مشهد، دانشگاه آزاد اسلامی، مشهد، ایران.

(تاریخ دریافت : ۱۳۹۸/۱۰/۱۶ تاریخ پذیرش : ۱۳۹۹/۷/۶)

چکیده

زمان‌بندی وظایف یکی از مهم‌ترین چالش‌ها در دستیابی به کارایی بالا در محیط‌های چندپردازشی مانند دستگاه‌های توزیع‌شده و موازی است. زمان‌بندی یک مسئله NP-سخت است که معمولاً روش‌های اکتشافی و فرا اکتشافی برای حل آن به کار گرفته می‌شوند. هدف از زمان‌بندی وظایف، اختصاص وظایف به منابع آزاد است به‌گونه‌ای که حداکثر موازی‌سازی در حین پردازش محقق شود. اکثر روش‌های موجود در ادبیات مبتنی بر راهکارهای ایستا و تک‌هدفه هستند. در این مقاله ابتدا یک مدل ریاضی از مسئله زمان‌بندی وظایف ارائه می‌شود. سپس زمان‌بندی پویای وظایف روی دستگاه‌های چندپردازشی ناهمگن با معیارهای ارزیابی متعدد موردمطالعه قرار می‌گیرد. درنهایت یک روش مبتنی بر الگوریتم فرا اکتشافی تبرید شبیه‌سازی‌شده هذلولی برای مسئله زمان‌بندی وظایف پیشنهاد می‌شود. ارزیابی‌های محاسباتی روی معیارهایی مانند زمان پردازش کل، نرخ همگرایی و زمان اجرا نشان‌دهنده عملکرد بالای الگوریتم پیشنهادی در مقایسه با روش‌های موجود در ادبیات برای مجموعه داده‌های یکسان است.

واژه‌های کلیدی: زمان‌بندی وظایف، الگوریتم تبرید شبیه‌سازی‌شده، نرخ همگرایی، منابع ناهمگن.

* **عهده‌دار مکاتبات:**

نشانی: گروه مهندسی کامپیوتر، واحد مشهد، دانشگاه آزاد اسلامی، مشهد، ایران.

پست الکترونیکی: *Drkamel@mshdiau.ac.ir*

۱. مقدمه

اصل محاسبات موازی برای استفاده بهینه از منابع سیستم و افزایش سرعت و کارایی پردازش برنامه روی پردازنده‌ها مطرح‌شده است. بخش‌هایی از برنامه اصلی که قابلیت اجرای هم‌زمان را دارند به چند زیر برنامه تقسیم و به‌صورت هم‌زمان روی چند منبع اجرا می‌شوند. بخشی از برنامه هم که قابلیت اجرای موازی ندارد به‌صورت ترتیبی روی یک منبع اجرا می‌شود [۱]. طبق قانون آمدال در محاسبات موازی، مقدار سرباری که به سیستم به دلیل تخصیص وظایف میان منابع تحمیل می‌شود باید کمتر از سود حاصل از موازی‌سازی باشد [۲]. درواقع تفاوت اصلی محاسبات ترتیبی و موازی همین موضوع می‌باشد که شرایط کاهش زمان محاسبه، امکان حل مسائل بزرگ‌تر، غلبه بر محدودیت‌های حافظه، صرفه‌جویی اقتصادی و استفاده از فناوری‌های روز مانند گرید و ابر را فراهم می‌کند. اما در پی آن مفاهیم متعددی مطرح می‌شوند که ماهیت آن‌ها با مفهوم متناظر در محاسبات معمولی کاملاً متفاوت است، مانند دانه‌بندی وظایف، محلی سازی، تعادل بار، همگام‌سازی و زمان‌بندی وظایف . در این مقاله به مسئله بهینه‌سازی زمان‌بندی وظایف پرداخته می‌شود.

مسئله زمان‌بندی وظایف یکی از مهم‌ترین چالش‌ها در دستیابی به کارایی بالا در محیط‌های چندپردازشی مانند دستگاه‌های توزیع‌شده و موازی است [۳]. این‌یک مسئله NP سخت است که معمولاً روش‌های اکتشافی و فرا اکتشافی برای حل آن بکار گرفته می‌شوند. هدف از زمان‌بندی وظایف، اولویت‌دهی و تخصیص وظایف به منابع آزاد است به‌گونه‌ای که حداکثر موازی‌سازی در حین پردازش محقق شود [۴]. اکثر روش‌های موجود در ادبیات مبتنی بر راهکارهای ایستا و تک‌هدفه هستند. روش‌های بهینه‌سازی رایج مبتنی بر گرادیان به دلیل وجود تعداد زیادی نقاط مینیمم محلی، وجود تعداد زیادی متغیر مستقل و یا دشواری محاسبه توابع هدف، ناکارآمد هستند [۵]. در مقابل، روش‌های بهینه‌سازی که وابسته به گرادیان نباشند، می‌توانند مقدار مینیمم سراسری تابع هدف مربوطه را پیداکرده و یک مدل مناسب از توزیع فضایی وظایف حاصل نمایند. یکی از معروف‌ترین روش‌های بهینه‌سازی غیر وابسته به گرادیان، روش تبرید شبیه‌سازی‌شده یا SA است. برخلاف روش‌های مبتنی بر گرادیان نزولی که تنها می‌توانند نقطه مینیمم نزدیک به حدس اولیه را پیدا کنند، روش تبرید شبیه‌سازی‌شده به نقطه مینیمم سراسری، همگرا شده و این همگرایی مستقل از حدس اولیه است [۵] [۶]. الگوریتم تبرید شبیه‌سازی‌شده، یک الگوریتم بهینه‌سازی فرا اکتشافی ساده و اثربخش در حل مسائل بهینه‌سازی در فضاهای جستجوی بزرگ است. این الگوریتم بیشتر زمانی استفاده می‌شود که فضای جستجو گسسته باشد. برای مسائلی که پیدا کردن یک پاسخ تقریبی برای بهینه‌سراسری مهم‌تر از پیدا کردن یک پاسخ دقیق برای بهینه محلی در زمان محدود و مشخص است، تبرید شبیه‌سازی‌شده نسبت به روش‌های دیگر دارای ارجحیت می‌باشد [۷].

کارهای عمده این مقاله به شرح ذیل بیان می‌شوند: ۱) ارائه یک مدل ریاضی از مسئله بهینه‌سازی زمان‌بندی وظایف ۲) توصیف زمان‌بندی پویای وظایفِ مستقل روی دستگاه‌های چندپردازشی ناهمگن با معیارهای ارزیابی متعدد (چندهدفه) ۳) ارائه یک روش فرا اکتشافی مبتنی بر الگوریتم تبرید شبیه‌سازی‌شده برای حل مسئله زمان‌بندی وظایف ۴) مقایسه روش پیشنهادی مقاله با نتایج حاصل از روش‌های زمان‌بندی مبتنی بر لیست.

ادامه مقاله به شرح ذیل سازمان‌دهی شده است: در بخش ۲ کارهای برجسته مرتبط با استفاده از الگوریتم‌های فرا اکتشافی در حل مسئله زمان‌بندی وظایف ارائه می‌شوند. بخش ۳ مسئله زمان‌بندی وظایف، مدل ریاضی آن و مفاهیم پایه بهینه‌سازی الگوریتم تبرید شبیه‌سازی‌شده توصیف می‌شود. بخش ۴ حاوی نتایج آزمایشی و تحلیل روش پیشنهادی است. درنهایت، نتیجه‌گیری و کارهای آینده در بخش ۵ ارائه می‌شوند.

۲- کارهای مرتبط

در سال‌های اخیر از الگوریتم‌های فرا اکتشافی متعددی برای حل مسائل زمان‌بندی استفاده‌شده است. اساساً مسائل زمان‌بندی که در ادبیات به آن‌ها پرداخته‌شده است را می‌توان بدون توجه به کاربرد در نظر گرفته‌شده به سه دسته زمان‌بندی وظیفه، زمان‌بندی کار و زمان‌بندی جریان‌های کاری تقسیم کرد. کاربردهای برجسته در این حوزه نیز محاسبات ابری، محاسبات گرید، دستگاه‌های چندپردازشی، خط تولید صنعتی و دستگاه‌های ماهواره‌ای هستند. در این بخش ادبیات منتشرشده در یک بازه زمانی ۴ ساله از ۲۰۱۷ تا ۲۰۲۰ پوشش داده می‌شود. تنها مقالات منتشرشده در مجلات ISI و Scopus در نظر گرفته‌شده‌اند. هر مقاله ازنظر ۶ شاخص موردبررسی قرار می‌گیرد که عبارت‌اند از ۱) حالت فضای مسئله (ایستا، پویا) ۲) زمان‌بندی بلادرنگ، ۳) تعداد اهداف ۴) نوع منابع ۵) کاربرد ۶) راه‌حل فرا اکتشافی (مبتنی بر تک جواب، مبتنی بر جمعیت). از این شاخص‌ها موارد ۲ و ۶ مربوط به راه‌حل مسئله و بقیه موارد مربوط به فضای مسئله هستند.

همان‌طور که در جدول ۱ مشاهده می‌شود ۱۲ مقاله از الگوریتم ژنتیک، ۴ مقاله از الگوریتم بهینه‌سازی ازدحام ذرات، ۳ مقاله از الگوریتم کلونی زنبورعسل، ۲ مقاله از الگوریتم کلونی مورچگان، ۲ مقاله از الگوریتم جستجوی ممنوعه، ۲ مقاله از الگوریتم تبرید شبیه‌سازی‌شده و یک مقاله از الگوریتم کوکو برای حل مسئله زمان‌بندی استفاده کرده‌اند. ۵۱.۸۵ درصد از مقالات بررسی‌شده منابع را به‌صورت ناهمگن و ۵۵.۵۵ درصد از مقالات، زمان‌بندی چندهدفه را در نظر گرفته‌اند. ۷۱.۰۷ درصد از مقالات فضای مسئله را ایستا در نظر گرفته‌اند یعنی زمان محاسباتی هر وظیفه روی هر منبع قبل از شروع زمان‌بندی مشخص است و توپولوژی گراف هرگز تغییر نمی‌کند. اگرچه در مسائل دنیای واقعی، پویایی شرایط و محیط، واقع‌بینانه‌تر است اما لحاظ کردن آن در مدل‌سازی مسئله پیچیدگی را به‌طور قابل‌توجهی افزایش می‌دهد. زمان‌بندی بلادرنگ نیز تنها در ۵ مقاله موردتوجه قرار گرفت که حاکی از یک حوزه تحقیقاتی مناسب برای مطالعات آینده می‌باشد. از روش‌های فرا اکتشافی برجسته‌ای که در سال‌های اخیر مطرح‌شده‌اند و در کاربردهای مختلفی اعمال‌شده‌اند مانند الگوریتم قهرمانی در تورنمنت، بهینه‌سازی ملهم از فیزیک نور، الگوریتم ریشه-پاجوش، الگوریتم قطره آب‌های هوشمند، الگوریتم گرگ‌های خاکستری و الگوریتم محاسبات دریا در این حوزه حداقل در زمان نگارش این مقاله هرگز استفاده‌نشده است.

جدول ۱. مقالات مرتبط با زمان‌بندی وظایف با روش‌های فرااکتشفافی منتشرشده از ۲۰۱۳ تا ۲۰۱۷

مرجع	نویسندگان	سال	فرا اکتشافی	جواب/جمعیت	کاربرد	منابع	اهداف	بلادرنگ بودن	فضای مسئله
۸	کونار و همکاران	۲۰۱۷	الگوریتم ژنتیک	جمعیت	چندپردازشی	ناهمگن	چندهدفه	✓	ایستا
۹	اکبری و همکاران	۲۰۱۷	الگوریتم ژنتیک	جمعیت	----	ناهمگن	چندهدفه	×	ایستا
۱۰	ژانگ و همکاران	۲۰۱۸	الگوریتم ژنتیک	جمعیت	محاسبات ابر	ناهمگن	تک‌هدفه	×	پویا
۱۱	عبدی و زرندی	۲۰۱۹	الگوریتم ژنتیک	جمعیت	چندپردازشی	ناهمگن	تک‌هدفه	×	ایستا
۱۲	فادل و همکاران	۲۰۱۹	الگوریتم ژنتیک	جمعیت	CPU/FPGA	همگن	تک‌هدفه	×	ایستا
۱۳	روپانتی و سالامی	۲۰۱۹	الگوریتم ژنتیک	جمعیت	چندپردازشی	همگن	چندهدفه	×	پویا

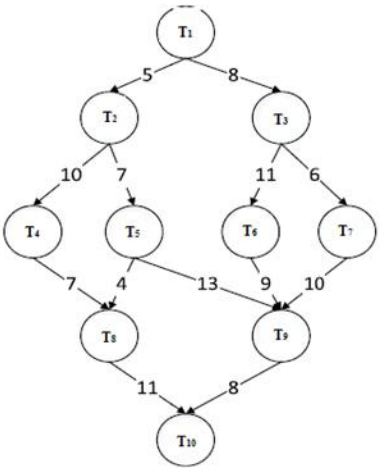
۱۴	وونگ و همکاران	۲۰۱۹	الگوریتم ژنتیک	جمعیت	سیستم های ماهواره	همگن	تک‌هدفه	×	پویا
۱۵	الگندی و همکاران	۲۰۱۹	الگوریتم ژنتیک	جمعیت	محاسبات ابر	ناهمگن	چندهدفه	×	پویا
۱۶	ماهوری و همکاران	۲۰۱۹	الگوریتم ژنتیک	جمعیت	چندپردازشی	همگن	تک‌هدفه	✓	ایستا
۱۷	ژانگ و همکاران	۲۰۱۹	الگوریتم ژنتیک	جمعیت	رادارهای آرایه	همگن	چندهدفه	×	پویا
۱۸	یان و همکاران	۲۰۱۹	الگوریتم ژنتیک	جمعیت	چندپردازشی روی تراشه	همگن	تک‌هدفه	✓	ایستا
۱۹	ولینگاری و همکاران	۲۰۲۰	الگوریتم ژنتیک	جمعیت	محاسبات ابر	ناهمگن	چندهدفه	×	پویا
۲۰	الهارکان و همکاران	۲۰۲۰	جستجوی ممنوعه	تک جواب	سرورهای موازی	همگن	تک‌هدفه	×	ایستا
۲۱	ولا و همکاران	۲۰۱۵	جستجوی ممنوعه (فازی)	تک جواب	کارگاهی	همگن	تک‌هدفه	×	پویا
۲۲	مرندی و همکاران	۲۰۱۹	تبرید شبیه‌سازی‌شده	تک جواب	شبکه چندفروشگاهی	ناهمگن	چندهدفه	×	ایستا
۲۳	رامش و همکاران	۲۰۲۰	تبرید شبیه‌سازی‌شده	تک جواب	فروشگاهی	ناهمگن	تک‌هدفه	×	پویا
۲۴	دردوایی و نجفی	۲۰۱۸	بهینه‌سازی ازدحام ذرات/تپه نوردی	جمعیت	محاسبات ابر	ناهمگن	تک‌هدفه	×	ایستا
۲۵	منصوری و همکاران	۲۰۱۹	بهینه‌سازی ازدحام ذرات (فازی)	جمعیت	محاسبات ابر	ناهمگن	چندهدفه	×	پویا
۲۶	بانسل و مالیک	۲۰۲۰	بهینه‌سازی ازدحام ذرات	جمعیت	محاسبات ابر	ناهمگن	چندهدفه	✓	پویا
۲۷	ژانگ و همکاران	۲۰۲۰	بهینه‌سازی ازدحام ذرات	جمعیت	محاسبات لبه	همگن	تک‌هدفه	×	پویا
۲۸	البود و کردی	۲۰۱۹	کوکو (فاخته)	جمعیت	کارگاهی	ناهمگن	چندهدفه	×	ایستا
۲۹	محمد کوردی	۲۰۱۹	کلونی مورچگان	جمعیت	فروشگاهی	ناهمگن	چندهدفه	×	ایستا
۳۰	نایی و همکاران	۲۰۲۰	کلونی مورچگان	جمعیت	سیستم سایبری/فیزیکی	ناهمگن	تک‌هدفه	×	ایستا
۳۱	شارما و همکاران	۲۰۱۸	کلونی زنبور مصنوعی	جمعیت	کارگاهی	ناهمگن	چندهدفه	✓	پویا
۳۲	ینگ لی و همکاران	۲۰۱۹	کلونی زنبور مصنوعی	جمعیت	فروشگاهی	ناهمگن	تک‌هدفه	×	پویا
۳۳	یی ینگ لی و همکاران	۲۰۲۰	کلونی زنبور مصنوعی	جمعیت	کارگاهی	ناهمگن	چندهدفه	×	ایستا

۳- روش پیشنهادی

در این بخش کاربرد و مدل سیستم توصیف می‌شود. هر کار از چندین وظایف تشکیل‌شده است و به‌عنوان یک گراف بدون دور جهت‌دار (DAG) مدل‌سازی می‌شود و با $G(V,E)$نشان داده می‌شود.

$V=\{T_1,T_2,\dots,T_n\}$مجموع رئوس نشان‌دهنده وظایف موجود در کار است و E مجموعه یال‌ها نشان‌دهنده وابستگی میان وظایف است. هر یال با یک برچسب نشان‌دهنده هزینه ارتباطی همراه شده است. نمونه‌ای از یک DAG در شکل ۱ نشان داده‌شده است. هزینه‌های محاسباتی وظایف موجود در شکل ۱ روی ۳ پردازنده در جدول ۱ ارائه‌شده است که در آن $\overline{w}$ هزینه محاسباتی میانگین T_i روی مجموع پردازنده‌هاست. وظیفه T_0 نشان‌دهنده گره ورودی و T_{10}نشان‌دهنده گره خروجی DAG است.

شکل ۱. نمونه‌ای از یک DAG با ۱۰ وظیفه



جدول ۲. هزینه محاسباتی وظایف روی پردازنده‌ها

Task	p_0	p_1	p_2	$\bar{w}$
T_1	9	10	8	9
T_2	12	11	13	12
T_3	15	8	10	11
T_4	6	10	8	8
T_5	15	13	11	13
T_6	7	9	14	10
T_7	13	17	15	15
T_8	12	10	14	12
T_9	18	13	17	16
T_{10}	7	10	7	8

۳-۱ تعریف مسئله

هزینه تکمیل هر وظیفه به‌صورت ذیل بیان می‌شود:

$$CTime_i = STime_i + ETime_i \quad i = 1, \dots, m \qquad (۱)$$

Makespan به‌عنوان کل زمان صرف‌شده برای تکمیل همه وظایف موجود بیان می‌شود:

$$Makespan = \sum_{i=1}^{m} Max\{CTime_i\} \qquad (۲)$$

هزینه پردازش هر وظیفه مجموع هزینه اجرا و هزینه انتقال داده‌های ورودی و خروجی است که به‌صورت ذیل بیان می‌شود:

$$PCost_i = ECost_i + InCost_i + OutCost_i \quad i = 1, \dots, m \qquad (۳)$$

توابع هدف مدل پیشنهادی به‌صورت ذیل تعریف می‌شود:

$$Min\ Z\_((1))\ (x) = Makespan \qquad (۴)$$

$$MinZ_{(2)}(x) = \sum_{j=1}^{n}\sum_{i=1}^{m} PCost_{ij} \times x_{ij}$$

$$subject\ to \begin{cases} \sum_{j=1}^{n} x_{ij=1,i=1,\dots,m} \\ x_{ij} \epsilon \{0,1\} \end{cases}$$

محدودیت اول یعنی هر وظیفه تنها روی یک منبع پردازش می‌شود.

۳-۲ الگوریتم تبرید شبیه‌سازی‌شده

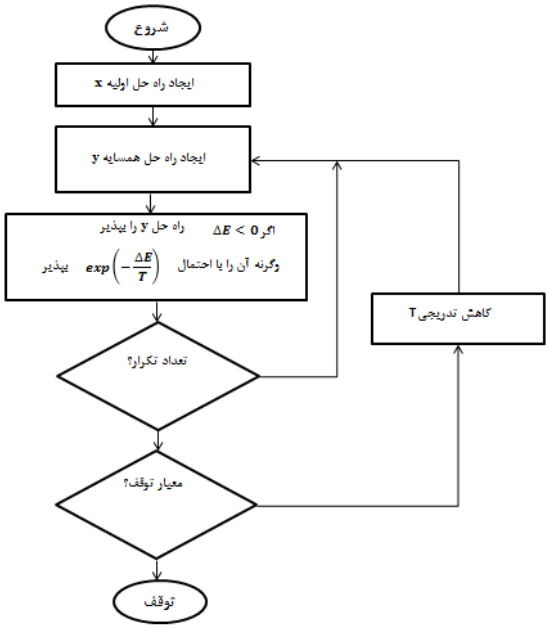
الگوریتم تبرید شبیه‌سازی‌شده یک الگوریتم فرا اکتشافی برای حل مسائلی است که دارای فضای جستجوی بزرگ و گسسته هستند [۲۳]. این الگوریتم الهام گرفته از فرآیند فیزیکی تبرید فلزات مذاب است. در فرآیند تبرید، یک فلزمذاب با دمای بسیار بالا به‌تدریج خنک می‌شود. در دماهای بالا، اتم‌های سازنده فلز به‌صورت تصادفی قرارگرفته‌اند و می‌توانند به‌آسانی نسبت به یکدیگر جابجا شوند. با کاهش تدریجی دما، حرکت اتم‌ها محدود می‌شود به‌طوری‌که اتم‌ها موقعیت خود را تثبیت می‌کنند و کریستال‌ها را تشکیل می‌دهند. هر چه سطح انرژی کریستال‌ها پایین‌تر باشد، از ساختار مطلوب‌تری برای استحکام فلز برخوردارند.

سطح انرژی کریستال تشکیل‌شده به‌سرعت سرد کردن فلز بستگی دارد. اگر کاهش دما خیلی سریع صورت گیرد، ممکن است ساختار غیر کریستالی با سطح انرژی بالا تشکیل شود. بنابراین برای رسیدن به کمترین میزان سطح انرژی، فرایند سرد کردن باید به‌آرامی صورت پذیرد. درواقع هدف نهایی تبرید، دست‌یابی به ساختار کریستالی با کمترین سطح انرژی به‌منظور دست‌یابی به استحکام فلز است. الگوریتم تبرید شبیه‌سازی‌شده به دنبال مقدار مینیمم سراسری یک تابع هدف است که معادل سطح انرژی فرایند تبرید می‌باشد [۳۴].

فرایند سردسازی با استفاده از یک پارامتر شبه دما شبیه‌سازی می‌شود. برای حل یک مسئله بهینه‌سازی، الگوریتم ابتدا از یک جواب اولیه s شروع می‌کند و سپس در یک حلقه تکرار به‌سوی جواب‌های همسایه حرکت می‌کند. اگر جواب همسایه بهتر از جواب فعلی باشد، الگوریتم آن را به‌عنوان جواب فعلی قرار می‌دهد (به‌سوی آن حرکت می‌کند)، در غیر این صورت، الگوریتم آن جواب را با احتمال پذیرش $P(e, e^*, T)$ به‌عنوان جواب فعلی می‌پذیرد که در آن $e = E(s)$ و $e^* = E(s^*)$ به ترتیب توابع انرژی جواب فعلی و جواب همسایه هستند و T یک پارامتر به نام دما است. در هر دما، چندین تکرار اجرا می‌شود و سپس دما به‌آرامی کاهش داده می‌شود. در گام‌های اولیه دما خیلی بالا قرار داده می‌شود تا احتمال بیشتری برای پذیرش جواب‌های بدتر وجود داشته باشد. با

کاهش تدریجی دما، در گام‌های پایانی احتمال کمتری برای پذیرش جواب‌های بدتر وجود خواهد داشت و بنابراین الگوریتم به سمت یک جواب خوب همگرا می‌شود. بنابراین در الگوریتم SA سه مؤلفه باید برای هر مسئله مشخص شود ۱) احتمال پذیرش ۲) تولید همسایه‌های جدید و ۳) زمان‌بندی کاهش دما که در بخش‌های ذیل به آن پرداخته می‌شود [۲۴]. نمودار جریان الگوریتم تبرید شبیه‌سازی‌شده در شکل ۲ نشان داده‌شده است.

شکل ۲. نمودار جریان الگوریتم تبرید شبیه‌سازی‌شده



۳-۲-۱ احتمال پذیرش

معمولاً در ادبیات رابطه‌ای که در فرمول ۶ نشان داده‌شده است به‌عنوان احتمال پذیرش استفاده می‌شود و به توزیع احتمالی بولتزمن مشهور است که در آن k نشان‌دهنده ثابت بولتزمن است. در توزیع مزبور شانس اینکه سیستم از یک مقدار انرژی مینیمم محلی بیرون آمده و به یک مقدار سراسری همگرا شود، وجود دارد. اگر جواب همسایه بهتر است آن را قبول می‌کنیم. اگر همسایه بهتر نباشد چند عامل را در نظر می‌گیریم. اولاً چه میزان همسایه بدتر است؟ دوما حرارت سیستم چقدر بالاست؟ در حرارت‌های بالا احتمال بیشتری وجود دارد که جواب‌های بدتر هم انتخاب شوند. بنابراین هر چه تغییر در انرژی کمتر (کیفیت جواب) و هر چه حرارت بیشتر باشد احتمال پذیرفتن جواب وجود دارد [۳۴].

$$P(e,e^*,T)\begin{cases}1 & if\ e^* \le e \\ exp(-(e^*-e)/kT) & otherwise\end{cases} \quad (۵)$$

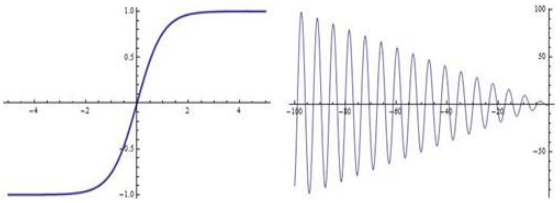
۳-۲-۲ تولید همسایه

تولید همسایه یا راه‌حل جدید معمولاً به‌صورت تصادفی و بر مبنای راه‌حل فعلی انجام می‌شود. البته در ادبیات روش‌های گوناگونی برای تابع تولید همسایه پیشنهادشده است که بنا به کاربرد تعریف‌شده متفاوت هستند. نحوه تعیین همسایه‌ها از جواب فعلی، روش ارزیابی همسایه‌ها و چگونگی انتخاب همسایه جایگزین جواب فعلی در هر روش فرا اکتشافی، به‌گونه‌ای خاص انجام می‌شود. در بین روال‌های بهینه‌یابی، روش‌های جواب یابی مکرر، نقش مهمی‌دارند. مهم‌ترین گام در یک روال تکراری عبارت است از تولید جواب جدید' s از جواب فعلی s و آزمون اینکه این گام باید تکرار شود یا نه. تکنیک‌های جستجوی همسایگی روال‌های تکراری هستند که در آن‌ها همسایه $N(s)$ برای هر جواب s تعریف می‌شود و جواب بعدی S^* از بین جواب‌های $N(s)$ انتخاب می‌شود [۳۵]. در این مقاله از روش تولید همسایه هذلولی (هایپربولیک) استفاده می‌شود (۱۴) که از دو تابع $xCos(x)$ و $tanh(x)$ استفاده می‌کند. تابع $xCos(x)$ دارای محور تقارن x =۰ است. شکل ۳(الف) این تابع را در بازه [۰ ۱۰۰-] نشان می‌دهد. با نزدیک شدن مقدار x به صفر، دامنه نوسان مقدار تابع کاهش می‌یابد که از این خاصیت برای همگرا نمودن الگوریتم استفاده‌شده است. تابع $tanh(x)$ دارای مرکز تقارن (۰ ۰) و برد (۱ ۱-) است. اگر عدد ثابتی مانند a در این تابع ضرب شود، برد آن شامل بازه باز $(-a,a)$ می‌شود که در شکل ۳(ب) نشان داده‌شده است. از این خاصیت برای تولید جواب‌هایی در فاصله اندک از جواب فعلی استفاده‌شده است [۳۶]. تابع تولید همسایه روش هذلولی در رابطه (۷) ذکر شده است:

$$S^* = 0.05\ \mathrm{Tanh}(\mathrm{iCos}(100\mathrm{i})) \times \mathrm{R} \times (\mathrm{U}-\mathrm{L}) + \mathrm{S} \quad (۷)$$

که در آن i عدد تکرار، R یک عدد تصادفی، U کران بالای جواب‌ها، L کران پایین جواب‌ها و S جواب فعلی است.

شکل ۳(الف) منحنی تابع xCos(x) در بازه [۰ ۱۰۰-]، (ب) منحنی تابع tanh(x)



۳-۲-۳ زمان‌بندی کاهش دما

زمان‌بندی کاهش دما یا استراتژی سردسازی سیستم، چگونگی کاهش دما را در حین تکرارهای روش تعیین می‌کند. کیفیت جواب SA به میزان زیادی بستگی به تعداد جواب‌هایی دارد که در هر دما موردبررسی قرار می‌گیرند. بنابراین مقدار این پارامتر هم‌بستگی به ابعاد و ساختار مسئله دارد و معمولاً با چند بار اجرای آزمایشی از طریق آزمون‌وخطا برای مقادیر مختلف تعیین می‌شود به‌گونه‌ای که در هر تکرار روش SA بایستی تعداد زیادی جواب تولید شوند تا یک تعادل حرارتی در هر دما برقرار گردد. نرخ سردسازی بسیار بزرگ، باعث همگرایی زودهنگام و حبس در مینیمم محلی می‌شود. نرخ سردسازی کوچک، زمان محاسباتی را افزایش می‌دهد. برای کاهش دادن دما، دمای فعلی را در ضریب α ضرب می‌کنیم. توجه کنید که α مقدار بین ۰ و ۱ است. در الگوریتم SA دما به‌تدریج و بسیار آهسته کم می‌شود، پس مقدار α باید به ۱ نزدیک باشد . کاهش دادن سریع دما باعث خواهد شد تا در مینیمم محلی گیر کند. بهترین مقداری که در ادبیات برای α پیشنهادشده است ۰.۹۷ است [۳۶].

۴- پیاده‌سازی

در فاز پیاده‌سازی کد برنامه در نرم‌افزار متلب R2014b v9.1 x64 نوشته شد و برنامه روی یک سیستم خانگی با پردازنده Intel® Core™2 Duo E4500 و ۲ گیگابایت حافظه RAM اجرا شد. منابع ناهمگن در نظر گرفته‌شده‌اند یعنی قابلیت آن‌ها برای پردازش وظایف مختلف متفاوت است و پردازش یک وظیفه روی منابع مختلف می‌تواند به زمان‌های اجرای متفاوتی منجر شود. هزینه ارتباط بین وظایف نیز در نظر گرفته‌شده است. هزینه ارتباط یا هزینه لبه در گراف متناظر یعنی زمانی که برای انتقال نتیجه یک وظیفه به یک وظیفه مجاور صرف می‌شود. پویایی زمان‌بندی به این معناست که پس‌ازاینکه هر وظیفه از صف حذف می‌شود و به یک منبع اختصاص داده می‌شود مجدداً تابع اولویت‌دهی اجرا می‌شود و چیدمانِ وظایف در صف را تغییر می‌دهد. اگرچه این وضعیت منجر به افزایش پیچیدگی محاسباتی می‌شود ولی قطعاً به کاهش زمان پردازش کل کمک می‌کند. HEFT یکی از الگوریتم‌های شناخته‌شده در این حوزه است که نتایج حاصل از آن در ادبیات به‌عنوان یک شاخص قیاسی برای نشان دادن عملکرد روش پیشنهادی این مقاله استفاده می‌شود. منطق HEFT بسیار ساده است و وظیفه را به منبع به‌گونه‌ای اختصاص می‌دهد که زودترین زمان اتمام آن ارضا شود. در بخش ۵، نتایج حاصل از پیاده‌سازی با HEFT مقایسه می‌شود. ضمن این‌که در تمام آزمایش‌ها از ماتریس‌های [۲۰×۴] تولیدشده تصادفی استفاده می‌شود.

۵- نتایج و بحث

در بخش اول، ابتدا نتایج حاصل از آزمایش‌ها روی معیار Makespan با توجه به مقادیر مختلف برای پارامترهای الگوریتم تبرید شبیه‌سازی‌شده بیان می‌شود. در بخش دوم نتایج سه معیار Makespan، زمان اجرای برنامه و نرخ همگرایی روش پیشنهادی ارائه و با نتایج الگوریتم HEFT مقایسه می‌شوند. همان‌طور که در بخش قبل توضیح داده شد، روش شبیه‌سازی تبرید دارای چهار پارامتر دمای اولیه، دمای نهایی، ضریب کاهش دما و تعداد تکرار الگوریتم در هر دما است. در این بخش به دنبال یافتن ترکیبی از این پارامترها هستیم که جواب بهتری را حاصل کند. برای این منظور مقدار هر یک از پارامترها را در بازه‌های مشخص تغییر می‌دهیم و نتایج حاصل از اجرای الگوریتم را برای یک گراف یکسان [۲۰×۴] از وظایف که در قالب ماتریس پیاده‌سازی شده است استخراج می‌کنیم. تعداد وظایف ۲۰ و تعداد پردازنده‌ها ۴ می‌باشد.

این نتایج در جدول ۳ ارائه‌شده‌اند. انتظار داریم که با افزایش دمای اولیه، افزایش ضریب کاهش دما و افزایش تعداد تکرار در هر دما نقاط بیشتری از فضای جواب مورد جست جو قرار بگیرند و زمان حل نیز افزایش یابد.

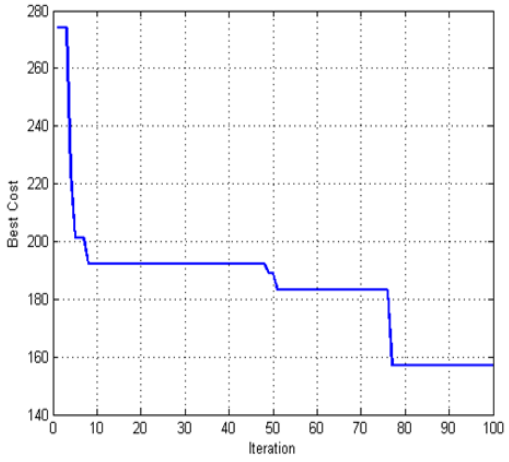
جدول ۳. مقادیر پارامترهای الگوریتم تبرید شبیه‌سازی‌شده

ردیف	دمای اولیه	ضریب کاهش دما	تعداد تکرار در هر دما	جواب نهایی الگوریتم	زمان اجرا
۱	۵۰	٪۹۷	۲۵	۱۵۹	۱۱.۲۰۶
۲	۷۵	٪۹۷	۲۵	۱۵۸	۱۱.۲۴۱
۳	۱۰۰	٪۹۷	۲۵	۱۵۸	۱۱.۳۳۲
۴	۱۲۵	٪۹۷	۲۵	۱۵۸	۱۱.۲۵۳
۵	۱۵۰	٪۹۷	۲۵	۱۵۸	۱۱.۳۸۸
۶	۱۷۵	٪۹۷	۲۵	۱۵۷	۱۱.۳۸۹
۷	۲۰۰	٪۹۷	۲۵	۱۶۴	۱۱.۴۰۴
۸	۲۲۵	٪۹۷	۲۵	۱۶۸	۱۱.۳۸۶
۹	۲۵۰	٪۹۷	۲۵	۱۶۶	۱۱.۴۰۸
۱۰	۳۰۰	٪۹۷	۲۵	۱۶۶	۱۱.۳۵۹

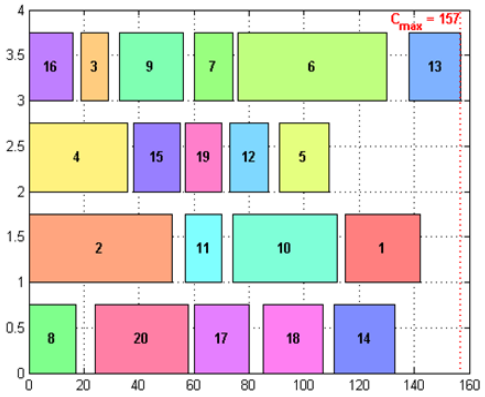
آزمایش با هر دمای اولیه ۳ بار انجام‌شده است و بدترین نتیجه در جدول ثبت‌شده است.

همان‌طور که در جدول مشخص است با افزایش دمای اولیه روند کلی به این صورت است که زمان اجرا افزایش می‌یابد اگرچه دمای اولیه ۳۰۰ زمان اجرای کمتری نسبت به دمای اولیه ۲۵۰ ثبت کرده است. افزایش بیشتر دمای اولیه تا ۱۷۵ موجب کاهش جواب نهایی الگوریتم شده است اما از این دما به بعد افزایش دمای اولیه لزوماً موجب بهتر شدن جواب‌ها نشده است. ضمن این‌که ضریب کاهش دما در تمام آزمایش‌ها به‌صورت ثابت ۰.۹۷ درصد در نظر گرفته‌شده است. این رایج‌ترین مقدار است که در ادبیات بکار گرفته‌شده است و عدد کوچک‌ترین از آن معمولاً منجر به حبس در بهینه‌های محلی می‌شود. بنابراین بر اساس نتایج به‌دست‌آمده، بهترین ترکیب پارامترها برای الگوریتم‌ها دمای اولیه ۱۷۵، تعداد تکرار در هر دما ۲۵، ضریب کاهش دمای ۰.۹۷ می‌باشد. نمودار همگرایی و Makespan این ترکیب پارامتریک به ترتیب در شکل ۴ و ۵ نشان داده‌شده است.

شکل ۴. نمودار همگرایی بهترین ترکیب پارامتریک: دمای اولیه ۱۷۵، تعداد تکرار در هر دما ۲۵ ، ضریب کاهش دمای ۰.۹۷



شکل ۵. نمودار Makespan بهترین ترکیب پارامتریک: دمای اولیه ۱۷۵، تعداد تکرار در هر دما۲۵ ، ضریب کاهش دمای ۰.۹۷

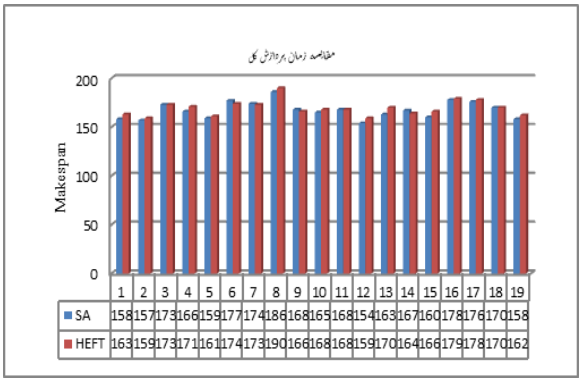


۱۷۵، تعداد تکرار در هر دما۲۵ ، ضریب کاهش دمای ۰.۹۷

در این بخش نتایج مقایسه الگوریتم تبرید شبیه‌سازی پیشنهادی با الگوریتم HEFT ارائه می‌شود. نمودار مقایسه ازلحاظ سه معیار Makespan، زمان پیاده‌سازی برنامه و نرخ همگرایی به ترتیب در شکل‌های ۶ و ۷ و ۸ نشان داده‌شده است. آزمایش روی یک گراف یکسان [۴×۲۰] در ۲۰ تکرار انجام‌شده است. هزینه پردازش وظایف روی پردازنده‌ها در بازه ۱۰ تا ۶۰ در نظر گرفته‌شده است.

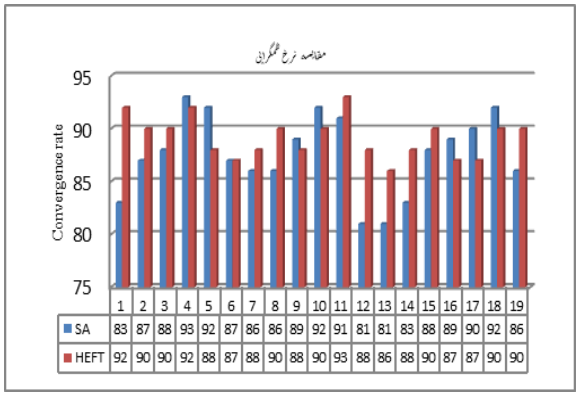
شکل ۶. مقایسه نتایج Makespan حاصل از الگوریتم‌های

SA و HEFT با ۲۰ گراف تصادفی (۲۰ وظیفه، ۴ پردازنده)



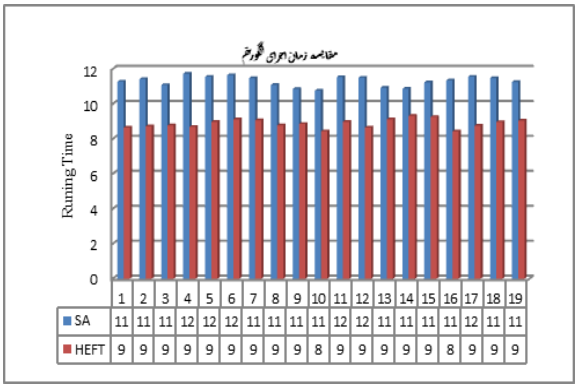
همان‌طور که در شکل ۶ نشان داده‌شده است، نتایج Makespan حاصل از الگوریتم SA در بازه ۱۵۴ و ۱۸۶ تغییر می‌کند و نتایج حاصل از الگوریتم HEFT بین ۱۵۹ و ۱۹۰ تغییر می‌کند. از ۲۰ تکرار آزمایش در ۱۶ تکرار زمان Makespan مربوط به SA کمتر یا برابر بازمان Makespan مربوط به HEFT است و تنها در تکرارهای ۶، ۷، ۹، ۱۴ زمان Makespan مربوط به HEFT کمتر از SA است که نشان‌دهنده برتری SA در ۸۰ درصد از موارد است.

شکل ۷. مقایسه نتایج نرخ همگرایی حاصل از الگوریتم‌های SA و HEFT در ۱۰۰ تکرار با ۲۰ گراف تصادفی



همان‌طور که در شکل ۷ نشان داده‌شده است، نتایج نرخ همگرایی حاصل از الگوریتم SA در بازه ۸۱ و ۹۳ تغییر می‌کند و نرخ همگرایی حاصل از الگوریتم HEFT بین ۸۶ و ۹۳ تغییر می‌کند. از ۲۰ تکرار آزمایش در ۱۴ تکرار نرخ همگرایی مربوط به SA کمتر یا برابر با نرخ همگرایی مربوط به HEFT است که نشان‌دهنده برتری SA در ۷۰ درصد از موارد است.

شکل ۸. مقایسه نتایج زمان پیاده‌سازی حاصل از الگوریتم‌های SA و HEFT در ۱۰۰ تکرار ۲۰ گراف تصادفی



و HEFT در ۱۰۰ تکرار ۲۰ گراف تصادفی

مجدداً همان‌طور که در شکل ۷ نشان داده‌شده است، نتایج زمان اجرای حاصل از الگوریتم SA در بازه ۱۰.۸۶ و ۱۱.۷۳ تغییر می‌کند و زمان اجرای حاصل از الگوریتم HEFT بین ۸.۴۴ و ۹.۳۲ تغییر می‌کند. در تمام ۲۰ تکرار آزمایش نتایج نشان‌دهنده برتری مطلق الگوریتم HEFT نسبت به الگوریتم SA است.

از برآیند نتایج مشاهده‌شده در بالا برتری الگوریتم SA نسبت به روش HEFT که یک روش پویای مبتنی بر لیست است ازنظر دو معیار نرخ همگرایی و زمان پردازش کل مشهود است. همچنین ازنظر معیار زمان اجرای برنامه نیز برتری مطلق روش HEFT نسبت به الگوریتم SA کاملاً آشکار است. راجع به برتری روش HEFT نسبت به روش‌های تکاملی مانند PSO، GA و GA-PSO ازنظر زمان اجرا در مقالات دیگر هم قبلاً صحبت شده است و اساساً اختلاف این نتایج همچنان یک معمای حل نشدنی است!

۶- نتیجه‌گیری

هدف از زمان‌بندی وظایف در دستگاه‌های توزیع‌شده، تخصیص وظایف به منابع ناهمگن است به‌گونه‌ای که برخی از معیارهای عملکرد، مانند حداقل زمان اجرا یا حداکثر سازی توازی بهبود یابند. زمان‌بندی وظایف یک

مسئله NP–سخت است، به همین دلیل از الگوریتم‌های اکتشافی و فرا اکتشافی برای حل آن استفاده می‌شود. در این مقاله یک الگوریتم SA با تابع تولید همسایه هذلولی برای زمان‌بندی چندهدفه پویای وظایف در محیط‌های پردازش موازی با منابع ناهمگن پیشنهادشده است. نتایج حاصل از اِعمال این الگوریتم روی DAG های مصنوعی با اندازه‌های مختلف نشان می‌دهد که روش پیشنهادی ازنظر نرخ همگرایی و زمان پردازش کل وظایف و میزان استفاده از پردازنده نسبت به روش HEFT مؤثرتر است. در کارهای آینده ما قصد داریم الگوریتم پیشنهادی را در قالب رویکردهای چندهدفه با در نظر گرفتن معیاری مانند میزان حرارت تولیدشده ناشی از عملیات پردازشی که اخیراً در مباحث محاسبات چندهسته‌ای مطرح‌شده است و همچنین در نظر گرفتن مؤلفه‌هایی از قبیل زمان‌بندی بلادرنگ و توازن بار پویا تعمیم دهیم. ضمن این‌که آزمون الگوریتم SA روی گراف‌های جهان واقعی در محیط‌های محاسبات ابری به‌عنوان یکی دیگر از کارهای تحقیقاتی آینده مطلوب خواهد بود.

مراجع

[1] Gottlieb, Allan, Almasi, George S, Highly parallel computing. Redwood City, Calif.: Benjamin/Cummings, 1989, pp. 21-23.

[2] Gustafson, John L, Reevaluating Amdahl's law, Communications of the ACM, 1988. pp. 323-346.

[3] Mehdi Akbaria, Hassan Rashidib, Sasan H. Alizadeh, "An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems", Engineering Applications of Artificial Intelligence, 2017, Vol. 61, pp. 35–46.

[4] Ullman, J.D., NP-complete scheduling problems. J. Comput. Syst. Sci. 10, 1979, pp. 384–393.

[5] Tran N. H. and Tran K., "Combination of fuzzy ranking and simulated annealing to improve discrete fracture inversion Elsevier", Mathematical and Computer Modeling, 2007, Vol. 45, pp. 1010–1020.

[6] Dudek G., "Adaptive simulated annealing schedule to the unit commitment problem", Elsevier, Electric Power Systems Research, 2010, Vol. 80, pp. 465–472.

[7] Cerny, V, "A thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm", Journal of Optimization Theory and Applications, 1985, Vol. 45, pp. 41–51.

[8] Debanjan Konara, Siddhartha Bhattacharyyab, Kalpana Sharmaa, Sital Sharmac, Sri Raj Pradhan, "An improved Hybrid Quantum-Inspired Genetic Algorithm (HQIGA) for scheduling of real-time task in multiprocessor system", Applied Soft Computing 2017, Vol. 53, pp. 296–307.

[9] Mehdi Akbaria, Hassan Rashidib, Sasan H. Alizadeh, "An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems", Engineering Applications of Artificial Intelligence, 2017, Vol. 61, pp. 35–46.

[10] Feifei Zhang, Jidong Ge, Zhongjin Li, Chuanyi Li, "A load-aware resource allocation and task scheduling for the emerging cloudlet system", Future Generation Computer Systems, 2018, Vol. 87, pp. 438-456.

[11] Athena Abdi, Hamid R.Zarandi, "A meta heuristic-based task scheduling and mapping method to optimize main design challenges of heterogeneous multiprocessor embedded systems", Microelectronics Journal, 2019, Vol. 87, pp. 1–11.

[12] Fadel Abdallah, Camel Tanougast, Imed Kacem, Camille Diou, "Genetic algorithms for scheduling in a CPU/FPGA architecture with heterogeneous communication delays", Computers & Industrial Engineering, 2019, Vol. 137, pp. 106-126.

[13] Dulana Rupanetti, Hassan Salamy, "Task allocation, migration and scheduling for energy-efficient real-time multiprocessor architectures", Journal of Systems Architecture, 2019, Vol. 98, pp. 17-26.

[14] Shuo Wang, Lin Zhao, Jianhua Cheng, Junfeng Zhou, Yipeng Wang, "Task scheduling and attitude planning for agile earth observation satellite with intensive tasks", Aerospace Science and Technology, 2019, Vol. 90, pp. 23-33.

[15] Abdelrahman Elgendy, Jihong Yan, Mingyang Zhang, "Integrated Strategies to an Improved Genetic Algorithm for Allocating and Scheduling Multi-Task in Cloud Manufacturing Environment", Procedia Manufacturing, 2019, Vol. 39, pp. 1872-1879.

[16] Pranab K.Muhuri, Amit Rauniyar, Rahul Nath, "On arrival scheduling of real-time precedence constrained tasks on multi-processor systems using genetic algorithm" Future Generation Computer Systems, 2019, Vol. 93, pp. 702-726.

[17] Zhang Haowei, Xie Junwei, Ge Jiaang, "A hybrid adaptively genetic algorithm for task scheduling problem in the phased array radar", European Journal of Operational Research, 2019, Vol. 272, pp. 868-878.
[18] YunYonghee, Hwang Eun Ju, Kim Young Hwan, "Adaptive genetic algorithm for energy-efficient task scheduling on asymmetric multiprocessor system-on-chip", Microprocessors and Microsystems, 2019, Vol. 66. pp. 19-30.
[19] S.Velliangiri, P.Karthikeyan, V.M.Arul Xavier, D.Baswaraj, "Hybrid electro search with genetic algorithm for task scheduling in cloud computing", Ain Shams Engineering Journal, 2020, (In press) Vol. XX, pp. XX–XX.
[20] Ibrahim Alharkan, Mustafa Saleh, Mageed A.Ghaleb, Husam Kaid, "Tabu search and particle swarm optimization algorithms for two identical parallel machines scheduling problem with a single server", Journal of King Saud University - Engineering Sciences, 2020, Vol. 32. pp. 330-338.
[21] Camino R.Vela, Sezin Afsar, Juan JoséPalacios, "Evolutionary tabu search for flexible due-date satisfaction in fuzzy job shop scheduling", Computers & Operations Research, 2020, Vol. 119. pp. 104–131.
[22] Fateme Marandi, Fatemi Ghomi, "Network configuration multi-factory scheduling with batch delivery: A learning-oriented simulated annealing approach", Computers & Industrial Engineering, 2019, Vol. 132, pp. 293-310.
[23] C. Ramesh, R. Kamalakannan, R. Karthik, C. Pavin, S. Dhivaharan, "A lot streaming based flow shop scheduling problem using simulated annealing algorithm", MaterialsToday: Proceeding, 2020, Vol. 7, pp. 58–85.
[24] Negar Dordaie Nima Jafari Navimipour, "A hybrid particle swarm optimization and hill climbing algorithm for task scheduling in the cloud environments", ICT Express, 2018, Vol. 4, pp. 199-202.
[25] Najme Mansouri, Behnam Mohammad Hasani Zade, Mohammad Masoud Javidi, "Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory", Computers & Industrial Engineering, 2019, Vol. 130, pp. 597-633.
[26] Mitali Bansal, Sanjay Kumar Malik, "A Multi-Faceted Optimization Scheduling Framework Based on the Particle Swarm Optimization Algorithm in Cloud Computing", Sustainable Computing: Informatics and Systems, 2020, In press, Vol. xx, pp. XX–XX.
[27] Yi Zhang, Yu Liu, Junlong Zhou, Jin Sun, Keqin Li, "Slow-movement particle swarm optimization algorithms for scheduling security-critical tasks in resource-limited mobile edge computing", Future Generation Computer Systems, 2020, 112, pp. 148-161.
[28] Ebtesam Aloboud, Heba Kurdi, "Cuckoo-inspired Job Scheduling Algorithm for Cloud Computing", Procedia Computer Science, 2019, Vol. 151, pp. 1078-1083.
[29] Mohamed Kurdi, "Ant colony system with a novel Non-Daemon Actions procedure for multiprocessor task scheduling in multistage hybrid flow shop", Swarm and Evolutionary Computation, 2019, Vol. 44. pp. 987-1002.
[30] NaYi, Jianjun Xu, Limei Yan, Lin Huang, "Task optimization and scheduling of distributed cyber–physical system based on improved ant colony algorithm", Future Generation Computer Systems, 2020, Vol. 109, pp. 134-148.
[31] Nirmala Sharma, Harish Sharma, Ajay Sharma, "Beer froth artificial bee colony algorithm for job-shop scheduling problem", Applied Soft Computing, 2018, pp. 507-524.
[32] Ying LiLi, Fan Li, Quan-Ke Pan, Liang Gao, "An Artificial Bee Colony Algorithm for the Distributed Hybrid Flowshop Scheduling Problem", Procedia Manufacturing, 2019, Vol. 39, pp. 1158-1166.
[33] Yibing Li, Weixing Huang, Rui Wu, Kai Guo, "An improved artificial bee colony algorithm for solving multi-objective low-carbon flexible job shop scheduling problem", Applied Soft Computing, 2020, Vol. 95, pp. 506-544.
[34] Kirkpatrick, S. , Gelatt, C. D. , Vecchi, M. P. , Optimization by simulated annealing, ScienceDirect, 1987, pp. 671–680.
[35] Ferland, J. A., Hertz, A., & Lavoie, A. (1996). "An Object-Oriented Methodology for Solving Assignment-Type Problems with Neighborhood Search Techniques", Operations Research, 1996, Vol. 44, pp. 347-359.
[36] Liu, J. "The impact of neighbourhood size on the process of simulated annealing: Computational experiments on the flowshop scheduling problem", 1999, Vol. 37, pp. 285-288.