مجله کنترل
ISSN (print) 2008-8345
ISSN (online) 2538-3752
جلد ۱۵، شماره ۲، تابستان ۱۴۰۰، صفحه ۹۶-۸۱

زمان‌بندی وظایف با استفاده از الگوریتم ترکیبی PSO-IWD در محیط‌های محاسبات ابری با منابع ناهمگن

علیرضا صادقی حصار[۱]، سیدرضا کامل[۲]، محبوبه هوشمند[۳]

[۱] گروه مهندسی کامپیوتر، واحد مشهد، دانشگاه آزاد اسلامی، مشهد، ایران، Alireza.Sadeghi89@yahoo.com

[۲] گروه مهندسی کامپیوتر، واحد مشهد، دانشگاه آزاد اسلامی، مشهد، ایران، Drkamel@mashdiau.ac.ir

[۳] گروه مهندسی کامپیوتر، واحد مشهد، دانشگاه آزاد اسلامی، مشهد، ایران، Houshmand@mshdiau.ac.ir

دریافت: ۱۳۹۷/۰۸/۰۶ ویرایش اول: ۱۳۹۸/۰۶/۰۹ ویرایش دوم: ۱۳۹۸/۱۲/۲۱ ویرایش سوم: ۱۳۹۹/۰۳/۰۲ پذیرش: ۱۳۹۹/۰۴/۰۷

چکیده: زمان‌بندی بهینه وظایف یکی از مهمترین چالش‌ها برای دست‌یابی به عملکرد مطلوب در محیط‌های توزیع‌شده مانند محاسبات ابری است. هدف از زمان‌بندی وظایف، تخصیص وظایف به منابع پردازشی است به‌گونه‌ای که برخی از معیارهای عملکردِ سیستم مانند زمان اجرا یا توازی بهینه شوند. زمان‌بندی وظایف یک مسئله NP-کامل است، از این رو از الگوریتم‌های اکتشافی یا فرااکتشافی برای حل آن استفاده می‌شود. چون ارائه‌دهندگان ابر، منابع محاسباتی را بر مبنای مدل «پرداخت به میزان استفاده» ارائه می‌کنند، الگوریتم زمان‌بندی وظایف بشدت هزینه کاربران در ابر را تحت تاثیر قرار می‌دهد. در این مقاله یک الگوریتم زمان‌بندی وظایف جدید بر اساس بهینه‌سازی ازدحام ذرات به‌عنوان یک روش فرااکتشافی پیشنهاد می‌شود که وظایف کاربران را به منابع آزاد در محیط‌های محاسبات ابری تخصیص می‌دهد. برای تقویت عملکرد روش بهینه‌سازی ازدحام ذرات از نظر سرعت همگرایی، الگوریتم قطره‌های آب هوشمند اِعمال می‌شود. نتایج اجرای این الگوریتم روی گراف‌های تصادفی، بهبود قابل توجه کاراییِ روش پیشنهادی در مقایسه با سایر الگوریتم‌های زمان‌بندی وظایف را نشان دادند.

کلمات کلیدی: محاسبات ابری، زمان‌بندی وظایف، بهینه‌سازی ازدحام ذرات، قطره آب‌های هوشمند، منابع ناهمگن.

Task Scheduling Using the PSO-IWD Hybrid Algorithm in Cloud Computing with Heterogeneous Resources

Alireza Sadeghi Hesar[1], Seyed Reza Kamel[2], Mahboobeh Houshmand[3]

Abstract: Optimal Task Scheduling is one of the most important challenges for achieving high performance in distributed environments such as cloud computing. The primary purpose of task scheduling is to allocate tasks to resources so that some of the system performance metrics will be optimized such as runtime or parallelism. Task scheduling is an NP-complete problem, so heuristic or metha-heuristic algorithms are used to solve it. Because cloud providers offer computing resources based on the pay-as-you-go model, the scheduling algorithm affects the users cost of the cloud. In this paper, a new cloud task scheduling algorithm based on particle swarm optimization as a metha-heuristic method is proposed that assigns users tasks to free resources in cloud computing environments. To enhance the convergence rate of the particle swarm optimization method, the intelligent water drops algorithm is applied. The results of this algorithm on random graphs showed a significant improvement in the performance of the proposed method compared to other task scheduling algorithms.

Keywords: Cloud Computing, Task Scheduling, Particle Swarm Optimization, Intelligent Water Drops, Heterogeneous Resources.

نویسنده عهده دار مکاتبات:سیدرضا کامل

[Downloaded from joc-isice.ir on 2026-11-14]
[DOR: 20.1001.1.20088345.1400.15.2.1.8]
[DOI: 10.52547/joc.15.2.81]

۱- مقدمه

محاسبات ابری یک محیط محاسباتی متداول است که شرایط دسترسی فراگیر به مجموعه مشترکی از منابع سیستمیِ قابل‌پیکربندی مانند شبکه‌ها، سرورها، فضای ذخیره‌سازی، برنامه‌های کاربردی و سرویس‌ها را فراهم می‌کند. این منابع می‌توانند بسرعت با حداقل فعالیت مدیریتی از طریق اینترنت در اختیار کاربران قرار گیرند [۱]. در محیط‌های توزیع‌شده بویژه ابر، ارائه سرویسِ برحسب تقاضا[1] با مقیاس‌پذیری و دسترس‌پذیری بالا یکی از مولفه‌های کلیدی است [۲]. بطور کلی، مصرف‌کنندگان محاسبات ابری مالک زیرساخت فیزیکی ابر نیستند و برای اجتناب از پرداخت هزینه‌های اضافی، آن را از ارائه‌دهندگان شخص ثالث اجاره می‌کنند. ارائه‌دهندگان سرویس‌های ابری معمولا مدل «پرداخت به میزان استفاده»[2] را بکار می‌گیرند. این مدلِ پرداخت، امکان پاسخگویی به نیازهای کوتاه مدت، عدم الزام به انعقاد قراردادهای بلندمدت و کاهش هزینه‌ها را تسهیل می‌کند [۳].

هدف از زمان‌بندی وظایف، تخصیص وظایف به منابع پردازشی است بگونه‌ای که برخی از معیارهای عملکردِ سیستم مانند زمان اجرا یا توازی بهینه شوند. مسئله زمان‌بندی وظایف در سیستم‌های پردازش توزیعی از جنبه‌های متفاوتی مانند ناهمگنی پردازشگرها، تحلیل کارایی و پیچیدگی‌های محاسباتی قابل بحث است. زمان‌بندی وظایف اساسا یک مسئله NP-کامل است [۴] که از الگوریتم‌های اکتشافی و فرااکتشافی برای حل آن استفاده می‌شود. الگوریتم‌های تکاملی مانند بهینه‌سازی ازدحام ذرات[3] یکی از انواع متداول روش‌های فرااکتشافی هستند که می‌توانند راه‌حل‌های مطلوب را در فضای جستجو کشف کنند. با این حال، زمان همگرایی ذرات به نقطه بهینه، طولانی است [۵].

در این مقاله ترکیب الگوریتم قطره‌های آب هوشمند[4] با بهینه‌سازی ازدحام ذرات برای افزایش سرعت جستجوی فضای مسئله و کاهش زمان همگرایی پیشنهاد می‌شود. هدف نهایی روش پیشنهادی طراحی یک الگوریتم بمنظور تخصیص بهینه وظایف به منابع ناهمگن در ابر از طریق افزایش توازی و کاهش زمان اجرای کل وظایف یا Make-Span است. الگوریتم پیشنهادی مزبور بدلیل این که صرفا برای مسئله زمان‌بندی وظایف در ابر ارائه می‌شود و برای مسائل بهینه‌سازی دیگر مورد آزمون قرار نمی‌گیرد یک الگوریتم بهینه‌سازی تک‌منظوره است.

کارهای عمده این مقاله به شرح ذیل بیان می‌شوند:

- ترکیب الگوریتم قطره‌های آب هوشمند با بهینه‌سازی ازدحام ذرات برای افزایش سرعت جستجوی فضای مسئله و کاهش زمان همگرایی
- کاهش زمان اجرای کل وظایف در محیط محاسبات ابری

در ادامه در بخش ۲ کارهای برجسته مرتبط با استفاده از الگوریتم‌های تکاملی در حل مسئله زمان‌بندی وظایف ارائه می‌شوند. بخش ۳ ابتدا مسئله زمان‌بندی وظایف کلاسیک با قیود پایه، مدل کلی آن و مبانی الگوریتم‌های بهینه‌سازی ازدحام ذرات و قطره‌های آب هوشمند را توصیف می‌کند و در نهایت به ارائه راه‌حل پیشنهادی می‌پردازد. متعاقبا در بخش ۴ جزئیات پیاده‌سازی راه‌حل پیشنهادی مورد بحث قرار می‌گیرد. بخش ۵ حاوی نتایج آزمایشی و تحلیل روش پیشنهادی است. در نهایت، نتیجه‌گیری و کارهای آینده در بخش ۶ ارائه می‌شوند.

۲- مرور ادبیات

معیارهای مختلفی را می‌توان برای طبقه‌بندی الگوریتم‌های فرااکتشافی اِعمال کرد مانند الگوریتم‌های مبتنی بر تک جواب و مبتنی بر جمعیت، الگوریتم‌های الهام گرفته شده از طبیعت[5] و بدون الهام از طبیعت، الگوریتم‌های با حافظه و بدون حافظه و در نهایت الگوریتم‌های قطعی و احتمالی [۶]. هدف این بخش، مرور مقالات مرتبط با زمان‌بندی از دیدگاه دسته‌بندی اول یعنی مبتنی بر یک جواب و مبتنی بر جمعیت است. الگوریتم‌های مبتنی بر یک جواب در حین فرایند جستجو یک جواب را تغییر می‌دهند، در حالی که در الگوریتم‌های مبتنی بر جمعیت در حین جستجو، یک جمعیت از جواب‌ها در نظر گرفته می‌شود. از الگوریتم‌های فرااکتشافی شناخته شده بر پایه جمعیت می‌توان به الگوریتم ژنتیک، برنامه‌ریزی ژنتیک، بهینه‌سازی کلونی مورچگان[6]، کلونی زنبورهای مصنوعی[7]، روش بهینه‌سازی ازدحام ذرات، الگوریتم قهرمانی در لیگ‌های ورزشی، الگوریتم شعله/پروانه[8] و الگوریتم قطره آب‌های هوشمند اشاره کرد. از الگوریتم‌های متداول فرااکتشافی مبتنی بر یک جواب می‌توان الگوریتم جستجوی ممنوعه و الگوریتم تبرید شبیه‌سازی شده[9] را نام برد. بدلیل افزایش روزافزون کاربرد الگوریتم‌های فرااکتشافی در حل مسائل بهینه‌سازی، در سال‌های اخیر مقالات مروری متعددی در حوزه زمان‌بندی وظایف منتشر شده‌اند مانند [۷]، [۸]، [۹] که خوانندگان علاقه‌مند می‌توانند به آن‌ها مراجعه کنند.

در این مطالعه یک فاصله زمانی ۷ ساله از ۲۰۱۲ تا ۲۰۱۹ پوشش داده می‌شود. فقط مقالات منتشر شده در پایگاه‌های استنادی اسکوپوس و ISI در نظر گرفته شده‌اند. هر مقاله از نظر ۶ شاخص مورد بررسی قرار می‌گیرد که عبارتند از: نوع الگوریتم فرااکتشافی اِعمال شده، پلت فرم یا محیطی که در آن مسئله مطرح شده‌است، نوع منابع (همگن، ناهمگن)، تعداد اهداف (تک، چندتایی)، زمان‌بندی بلادرنگ و فضای مسئله (ایستا، پویا). خلاصه‌ای از مقالات بررسی شده در جدول ۱ ارائه شده است.

[1] On-demand
[2] Pay-as-you-go
[3] Particle Swarm Optimization
[4] Intelligent Water Drops
[5] Nature Inspired Algorithm
[6] Ant Colony Optimization
[7] Artificial Bee Colony
[8] Moth-Flame Optimization
[9] Simulated Annealing

[Downloaded from joc-isice.ir on 2026-11-14]
[DOR: 20.1001.1.20088345.1400.15.2.1.8]
[DOI: 10.52547/joc.15.2.81]

همانطور که در جدول ۱ مشاهده می‌شود، در میان روش‌های مبتنی بر جمعیت، الگوریتم ژنتیک و در میان روش‌های تک جوابی الگوریتم تبرید شبیه‌سازی شده، روش‌های غالب هستند. ۵۵.۸۸ درصد از مقالات منابع ناهمگن را در نظر گرفته‌اند. تمامی مقالاتی که زمان‌بندی در سیستم‌های چندپردازشگر را مطالعه کرده‌اند پیش‌فرضِ منابع همگن را لحاظ کرده‌اند [۱۰]، [۲۱]، [۳۵]، [۳۹]، [۴۰]. مقالاتی که پلت‌فرم‌های کارگاهی و فروشگاهی را برای زمان‌بندی وظایف انتخاب کرده‌اند منابع را ناهمگن در نظر گرفته‌اند که به واقعیت نزدیک‌تر است [۲۵]، [۲۸]، [۳۱]. ۵۵.۸۲ درصد از مقالات، زمان‌بندی چندهدفه را در نظر گرفته‌اند. از میان اهداف در نظر گرفته شده برای بهینه‌سازی می‌توان به میانگین Make-Span، سرعت همگرایی، میانگین زمان اجرای الگوریتم، نسبت موفقیت ((تعداد وظایف زمان‌بندی شده / تعداد کل وظایف)، میانگین نسل‌های همگرایی (میانگین تعداد نسل‌های مورد نیاز در هر تکرار از شبیه‌سازی برای دست‌یابی به راه‌حل بهینه) اشاره کرد. میانگین Make-Span تنها معیار هدفی است که در تمامی مقالات مد نظر بوده است. بعد از آن به ترتیب سرعت همگرایی در ۱۳ مقاله، میانگین زمان اجرای الگوریتم در ۸ مقاله، میانگین نسل‌های همگرایی در ۸ مقاله، میانگین زمان اجرای الگوریتم در ۴ مقاله و نسبت موفقیت در ۱ مقاله مورد مطالعه قرار گرفته‌اند.

همانطور که در جدول ۱ مشاهده می‌شود، در میان روش‌های مبتنی بر جمعیت، الگوریتم ژنتیک و در میان روش‌های تک جوابی الگوریتم تبرید شبیه‌سازی شده، روش‌های غالب هستند. ۵۵.۸۸ درصد از مقالات منابع ناهمگن را در نظر گرفته‌اند. تمامی مقالاتی که زمان‌بندی در سیستم‌های چندپردازشگر را مطالعه کرده‌اند پیش‌فرضِ منابع همگن را لحاظ کرده‌اند [۱۰]، [۲۱]، [۳۵]، [۳۹]، [۴۰]. مقالاتی که پلت‌فرم‌های کارگاهی و فروشگاهی را برای زمان‌بندی وظایف انتخاب کرده‌اند منابع را ناهمگن در نظر گرفته‌اند که به واقعیت نزدیک‌تر است [۲۵]، [۲۸]، [۳۱].

۵۵.۸۲ درصد از مقالات، زمان‌بندی چندهدفه را در نظر گرفته‌اند. از میان اهداف در نظر گرفته شده برای بهینه‌سازی می‌توان به میانگین Make-Span، سرعت همگرایی، میانگین زمان اجرای الگوریتم، نسبت موفقیت ((تعداد وظایف زمان‌بندی شده / تعداد کل وظایف)، میانگین نسل‌های همگرایی (میانگین تعداد نسل‌های مورد نیاز در هر تکرار از شبیه‌سازی برای دست‌یابی به راه‌حل بهینه) اشاره کرد. میانگین Make-Span تنها معیار هدفی است که در تمامی مقالات مد نظر بوده است. بعد از آن به ترتیب سرعت همگرایی در ۱۳ مقاله، میانگین زمان اجرای الگوریتم در ۸ مقاله، میانگین نسل‌های همگرایی در ۸ مقاله، میانگین زمان اجرای الگوریتم در ۴ مقاله و نسبت موفقیت در ۱ مقاله مورد مطالعه قرار گرفته‌اند.

لازم به ذکر است برخی از مقالات که روش تحقیق خود را برای یک مورد کاربردی خاص طراحی کرده‌اند از برخی اهداف مختصِ کاربرد بهره برده‌اند مثل میانگین تاخیر در خط تولید [۳۱]، [۴۰]، تاخیر انتها به انتها در داده‌های جریانی [۱۷]، [۲۷] و حرارت تولید شده در هسته‌ها [۲۹]. ۶۱.۷۶ درصد از مقالات فضای مسئله را ایستا در نظر گرفته‌اند یعنی ترتیب اجرای وظایف در صف با ورود وظایف جدید تغییر نمی‌کند و متعاقبا توپولوژی گراف هم هرگز تغییر نخواهد کرد. اگر چه در مسائل دنیای واقعی، پویایی شرایط و محیط، واقع‌بینانه‌تر است اما لحاظ کردن آن در مدل‌سازی مسئله پیچیدگی را بطور قابل توجهی افزایش می‌دهد.

همچنین در نظر گرفتن همزمان پویایی زمان‌بندی و منابع ناهمگن تنها در ۱۱ مقاله مورد مطالعه قرار گرفته‌است [۱۱]، [۱۳]، [۱۴]، [۱۵]، [۱۷]، [۱۸]، [۱۹]، [۲۴]، [۳۴]، [۳۷]، [۳۸] که حاکی از دشواری مدل‌سازی همزمان دو قید می‌باشد. زمان‌بندی بلادرنگ نیز تنها در ۳ مقاله [۱۰]، [۲۷] [۳۵]، مورد توجه قرار گرفته است که دو مورد از آن‌ها مرتبط با زمان‌بندی در سیستم‌های چندپردازشی و یک مورد مرتبط با حسگرهای رادار است و در هر سه مقاله کاربرد در نظر شده توسط نویسندگان یک کاربرد زمان-بحرانی معرفی شده‌است. در سال‌های اخیر اغلب زمان‌بندی بلادرنگ در کاربردهای مسیریابی، تشخیص پزشکی و امداد و نجات مطرح شده‌است و حاکی از یک حوزه تحقیقاتی مناسب برای مطالعات آینده می‌باشد.

از آن جایی که راه‌حل پیشنهادی در این مقاله ترکیبی از بهینه‌سازی ازدحام ذرات و قطره‌های آب هوشمند است پس یک راه حل فرااکتشافی مبتنی بر جمعیت خواهد بود. پلت فرم مورد نظر، محیط محاسبات ابری است. بنابراین طبق ماهیت محاسبات ابری، منابع در مسئله تحت بررسی ناهمگن هستند. فضای مسئله پویاست چون اولویت وظایف زمان‌بندی شده با گذشت زمان تغییر می‌کند. همچنین طبق آمار استخراج شده در جدول ۱ معیارهای میانگین Make-Span و سرعت همگرایی را بعنوان اهداف مقاله خود انتخاب می‌کنیم که مکررترین اهداف موجود در ادبیات هستند.

زمان‌بندی این تحقیق بلادرنگ نخواهد بود زیرا کاربرد مورد نظر در این مقاله یک کاربرد زمان-بحرانی نیست و بلادرنگ بودن زمان‌بندی، اولویت محسوب نخواهد شد.

[Downloaded from joc-isice.ir on 2026-11-14]
[DOR: 20.1001.1.20088345.1400.15.2.1.8]
[DOI: 10.52547/joc.15.2.81]

جدول ۱: مقالات مرتبط با زمان‌بندی وظایف با روش‌های فرااکتشافی منتشر شده از ۲۰۱۲ تا ۲۰۱۹

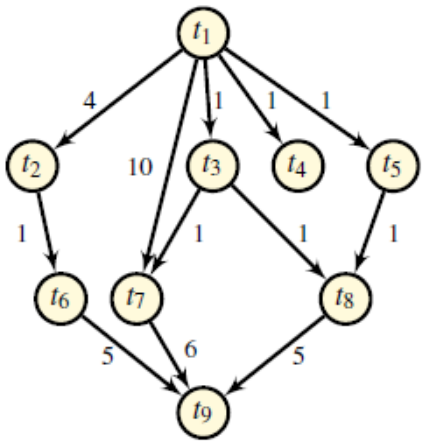
ردیف	شماره مرجع	سال	راه‌حل مورد استفاده	مبتنی بر تک جواب/جمعیت	پلت فرم	منابع	تعداد اهداف	بلادرنگ	ایستا/پویا
۱	[۱۰]	۲۰۱۹	الگوریتم ژنتیک	جمعیت	چندپردازشی	همگن	چندهدفه	✓	پویا
۲	[۱۱]	۲۰۱۹	الگوریتم ژنتیک + جستجوی هارمونی	جمعیت	ابر	ناهمگن	چندهدفه	×	پویا
۳	[۱۲]	۲۰۱۹	جستجوی ممنوعه + بهینه‌سازی ازدحام ذرات	جمعیت	ماشین‌های موازی	همگن	تک‌هدفه	×	ایستا
۴	[۱۳]	۲۰۱۹	بهینه‌سازی ازدحام ذرات	جمعیت	ابر	ناهمگن	چندهدفه	×	پویا
۵	[۱۴]	۲۰۱۹	بهینه‌سازی ازدحام ذرات + تئوری فازی	جمعیت	ابر	ناهمگن	تک‌هدفه	×	پویا
۶	[۱۵]	۲۰۱۹	شعله/پروانه + تکامل تفاضلی	جمعیت	ابر	ناهمگن	چندهدفه	×	پویا
۷	[۱۶]	۲۰۱۸	الگوریتم ژنتیک	جمعیت	ابر	ناهمگن	چندهدفه	×	ایستا
۸	[۱۷]	۲۰۱۸	تبرید شبیه‌سازی‌شده	تک جواب	مشاهدات ماهواره	ناهمگن	چندهدفه	×	پویا
۹	[۱۸]	۲۰۱۷	الگوریتم ژنتیک	جمعیت	ابر	ناهمگن	تک‌هدفه	×	پویا
۱۰	[۱۹]	۲۰۱۷	الگوریتم ژنتیک	جمعیت	ابر	ناهمگن	تک‌هدفه	×	پویا
۱۱	[۲۰]	۲۰۱۷	الگوریتم ژنتیک	جمعیت	ابر	ناهمگن	چندهدفه	×	ایستا
۱۲	[۲۱]	۲۰۱۷	کلونی مورچگان	جمعیت	چندپردازشی	همگن	تک‌هدفه	×	ایستا
۱۳	[۲۲]	۲۰۱۷	کلونی مورچگان	جمعیت	ابر	همگن	تک‌هدفه	×	ایستا
۱۴	[۲۳]	۲۰۱۷	کلونی زنبور مصنوعی	جمعیت	ابر	ناهمگن	چندهدفه	×	ایستا
۱۵	[۲۴]	۲۰۱۷	بهینه‌سازی وال	جمعیت	ابر	ناهمگن	چندهدفه	×	پویا
۱۶	[۲۵]	۲۰۱۷	بهینه‌سازی وال	جمعیت	کارگاه تولیدی	ناهمگن	چندهدفه	×	ایستا
۱۷	[۲۶]	۲۰۱۶	الگوریتم ژنتیک + بهینه‌سازی ازدحام ذرات	جمعیت	سیستم‌های سرویس‌گرا	همگن	چندهدفه	×	ایستا
۱۸	[۲۷]	۲۰۱۶	جستجوی ممنوعه	تک جواب	حسگرهای رادار	همگن	چندهدفه	✓	ایستا
۱۹	[۲۸]	۲۰۱۶	بهینه‌سازی ازدحام ذرات	جمعیت	سیستم‌های سرویس‌گرا	ناهمگن	چندهدفه	×	ایستا
۲۰	[۲۹]	۲۰۱۶	کوکو (فاخته)	جمعیت	پردازنده‌های موازی	ناهمگن	چندهدفه	×	ایستا
۲۱	[۳۰]	۲۰۱۵	الگوریتم ژنتیک	جمعیت	گرید	ناهمگن	چندهدفه	×	ایستا
۲۲	[۳۱]	۲۰۱۵	الگوریتم ژنتیک	جمعیت	کارگاه تولیدی	ناهمگن	تک‌هدفه	×	ایستا
۲۳	[۳۲]	۲۰۱۵	جستجوی ممنوعه	تک جواب	---	همگن	چندهدفه	×	ایستا
۲۴	[۳۳]	۲۰۱۵	تبرید شبیه‌سازی‌شده	تک جواب	ابر	ناهمگن	چندهدفه	×	ایستا
۲۵	[۳۴]	۲۰۱۴	الگوریتم ژنتیک	جمعیت	----	ناهمگن	چندهدفه	×	پویا
۲۶	[۳۵]	۲۰۱۳	الگوریتم ژنتیک	جمعیت	چندپردازشی	همگن	چندهدفه	✓	پویا
۲۷	[۳۶]	۲۰۱۳	الگوریتم ژنتیک	جمعیت	----	همگن	تک‌هدفه	×	ایستا
۲۸	[۳۷]	۲۰۱۳	تبرید شبیه‌سازی‌شده	تک جواب	سیستم‌های روی تراشه	ناهمگن	چندهدفه	×	پویا
۲۹	[۳۸]	۲۰۱۳	بهینه‌سازی ازدحام ذرات	جمعیت	ابر	ناهمگن	تک‌هدفه	×	پویا
۳۰	[۳۹]	۲۰۱۳	بهینه‌سازی ازدحام ذرات	جمعیت	چندپردازشی	همگن	تک‌هدفه	×	ایستا
۳۱	[۴۰]	۲۰۱۲	کلونی زنبور مصنوعی	جمعیت	چندپردازشی	همگن	تک‌هدفه	×	ایستا

۳- روش پیشنهادی

این بخش با معرفی مسئله زمان‌بندی آغاز می‌شود و در ادامه الگوریتم پیشنهادی ارائه خواهد شد. هر کار بعنوان یک گراف بدون دور جهت‌دار (DAG) مدل‌سازی می‌شود و با $G(V, E)$ نشان داده می‌شود. مجموعه گره‌ها $V = \{T_1, T_2, \dots, T_n\}$ نشان‌دهنده وظایف موجود در کار است و مجموعه لبه‌ها نشان‌دهنده وابستگی‌های کنترل/داده میان وظایف است. لبه $(T_i, T_j) \in E$ به این معنی است که داده‌های گره T_i توسط گره T_j مصرف می‌شوند. تا زمانی که T_i تکمیل نشده است T_j آغاز نمی‌شود. لبه‌ها با هزینه ارتباط برچسب‌گذاری شده‌اند. نمونه‌ای از یک DAG با ۹ وظیفه و ۱۲ لبه

[Downloaded from joc-isice.ir on 2026-11-14]
[DOR: 20.1001.1.20088345.1400.15.2.1.8]
[DOI: 10.52547/joc.15.2.81]

در شکل ۱ نشان داده شده است. هزینه محاسباتی وظایف موجود در شکل ۱ روی ۳ پردازنده در جدول ۲ ارائه شده است که در آن m_i، نشان‌دهنده i امین پردازنده و $\bar{w}$ هزینه محاسباتی میانگینِ T_i روی مجموع پردازنده‌هاست. وظیفه T_1 نشان‌دهنده وظیفه متعلق به گره ورودی و T_9 نشان‌دهنده وظیفه متعلق به گره خروجی DAG است. زمان‌بندی معتبر است اگر و تنها اگر چهار شرط برآورده شوند: ۱) تمام وظایف زمان‌بندی شوند ۲) هر وظیفه قبل از اجرا تمام اطلاعات مورد نیاز خود را دریافت کند ۳) هر وظیفه در هر زمان تنها روی یک منبع اجرا شود ۴) هر منبع در هر زمان تنها یک وظیفه را اجرا کند.



شکل ۱: نمونه‌ای از DAG پیشنهادی [۳۰].

Tasks	m_1	m_2	m_3	$\bar{w}$
t_1	3	1	2	2
t_2	3	3	3	3
t_3	4	3	2	3
t_4	4	2	6	4
t_5	7	5	3	5
t_6	2	6	4	4
t_7	4	6	2	4
t_8	4	3	5	4
t_9	1	1	1	1

جدول ۲: هزینه محاسباتی وظایف روی پردازنده‌ها [۳۰]

۱-۳ بهینه‌سازی ازدحام ذرات

الگوریتم بهینه‌سازی ازدحام ذرات (PSO) اولین بار توسط کندی و ابرهارت[10] [۴۱] مطرح شد. الگوریتم PSO یک الگوریتم جستجوی اجتماعی است که از رفتار گروهی دسته‌های پرندگان الهام گرفته شده‌است. مانند سایر الگوریتم‌های مبتنی بر جمعیت، الگوریتم PSO از یک مجموعه پاسخ‌های ممکن استفاده می‌کند که این پاسخ‌ها تا زمانی‌که یک پاسخ بهینه یافت شود و یا شرایط پایان الگوریتم محقق شود به حرکت خود ادامه می‌دهند. در این روش هر پاسخ به صورت یک ذره نمایش داده می‌شود. سردسته بهترین موقعیت را دارد. هر ذره در حین جستجو برای دست‌یابی به نقطهٔ بهینه دائما جابجا می‌شود. به دلیل این جابجایی، ذره دارای سرعت است. در این روش معادله سرعت، ضامن حرکت ذرات به سمت ناحیه بهینه است. این معادله براساس چهار عنصر اصلی ارائه می‌شود که عبارتند از (الف) مولفه محلی p-best (بهترین وضعیتی که ذره تاکنون تجربه کرده است)، (ب) مولفه جمعی g-best (بهترین ذره در میان ذرات)، (ج) سرعت و (د) مکان که از طریق روابط (۱) و (۲) محاسبه می‌شوند [۴۱]، [۴۲].

$$V_i(t+1) = wV_i(t) + C_1R_{1,i}(t)\big(P_i(t) - X_i(t)\big) + C_2R_{2,i}(t)\left(P_g(t) - X_i(t)\right) \quad (۱)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (۲)$$

که در آن $V_i(t+1)$ سرعت آینده ذره، $V_i(t)$ سرعت فعلی ذره، $X_i(t+1)$ مکان بعدی ذره، $X_i(t)$ مکان فعلی ذره، w وزن اینرسی، C_1 و C_2 مقادیر ثابت و مثبت و R_1 و R_2 اعداد تصادفی هستند که به صورت نرمال در بازه [۰،۱] تولید می‌شوند.

C_1 مقدار اهمیت مربوط به بهترین ذره و C_2 مقدار اهمیت مربوط به بهترین همسایگی‌ها است. به دلایل تجربی معمولاً $C_1 + C_2 = 4$ درنظر گرفته می‌شود. برای قابلیت بهتر جستجو، پارامتری به نام وزن اینرسی به صورت ضریب در معادله سرعت الگوریتم اضافه می‌گردد. معمولا الگوریتم با مقدار بزرگی از وزن اینرسی شروع به حرکت می‌کند که سبب جستجوی گسترده فضا در ابتدای اجرای الگوریتم شده و این وزن به مرور در طول زمان کاهش می‌یابد که سبب تمرکز جستجو در فضای کوچک در گام‌های پایانی می‌شود [۴۱]، [۴۲]. در ابتدا ذرات به صورت تصادفی در سرتاسر فضای جستجو مقداردهی می‌شوند که این موقعیت‌های اولیه به عنوان بهترین تجربه شخصی ذرات نیز شناخته می‌شوند (p-best). در گام بعد بهترین ذره از میان ذرات موجود انتخاب و بعنوان بهترین پاسخ شناخته

[10] Kennedy and Eberhart

[Downloaded from joc-isice.ir on 2026-11-14]
[DOR: 20.1001.1.20088345.1400.15.2.1.8]
[DOI: 10.52547/joc.15.2.81]

می‌شود (g-best). سپس گروه ذرات در فضای جستجو حرکت می‌نمایند، تا زمانی که شرایط پایانی محقق شود. این حرکت شامل اِعمال معادله سرعت به گروه ذرات می‌باشد که موقعیت هر ذره براساس آن تغییر می‌کند. مقدار برازش جدید حاصل از ذره، با مقدار p-best ذره مقایسه می‌گردد. اگر موقعیت جدید ذره دارای برازش بهتری باشد، این موقعیت جدید جایگزین موقعیت p-best می‌شود. روالی مشابه نیز برای g-best انجام می‌گیرد [۴۲]، [۴۳].

۲-۳ الگوریتم قطره‌های آب هوشمند

الگوریتم قطره‌های آب هوشمند (IWE) در سال ۲۰۰۹ توسط شاه حسینی[11] [۴۴] بر اساس رفتار جریان آب ارائه شد. قطرات آب موجود در رودخانه‌ها به طور هوشمندانه کوتاه‌ترین مسیر در جهت رسیدن به دریا را پیدا می‌کنند. در این الگوریتم دو ویژگی مهم سرعت و میزان خاک دریافتی از زمین برای قطرات تعریف شده است. هرچه میزان خاک دریافتی کمتر باشد سرعت قطرات بیشتر می‌شود. میزان خاک در واقع متناظر با اطلاعاتی است که بین زمین و قطره‌های آب مبادله می‌شود. اساسا یک قطره مسیری را انتخاب می‌کند که مجبور باشد خاک کمتری را در حین پیمایش آن حمل کند. اگر گام انتقال قطره‌ها بصورت گسسته در نظر گرفته شود می‌توان فرایند جابجایی قطره‌ها را روی گره‌های یک گراف نگاشت کرد. هر قطره آب یا IWD نماینده یکی از پاسخ‌های مسئله است و بصورت تصادفی روی هر یک از گره‌های گراف مقداردهی می‌شود. هر IWD دارای یک لیست گره بازدید شده $V_c(IWD)$ است که در ابتدا تهی در نظر گرفته می‌شود. در هر تکرار از الگوریتم مراحل ذیل برای هر قطره انجام می‌شود.

مرحله ۱. به ازای هر قطره IWD ساکن در گره i، گره بعدی j با احتمال E بگونه‌ای انتخاب می‌شود که در لیست گره‌های بازدید شده IWD نباشد و ضمنا شروط مسئله را نقض نکند. متغیر E متناسب با معکوس میزان خاک موجود بین دو گره است و طبق رابطه (۳) محاسبه می‌شود.

$$E_i^{IWD}(j) = \frac{f\big(soil(i,j)\big)}{\sum_{k\notin vc(IWD)} f\big(soil(i,k)\big)} \qquad (۳)$$

که در آن K گره‌های میانی بین دو گره i و j است و $f\big(soil(i,j)\big)$ از رابطه (۴) محاسبه می‌شود.

$$f\big(soil(i,j)\big) = \frac{1}{g\big(soil(i,j)\big)} \qquad (۴)$$

و متعاقبا $g\big(\mathrm{soil}(i,j)\big)$ طبق رابطه (۵) محاسبه می‌شود.

$$g\big(soil(i,j)\big) = \begin{cases} soil(i,j) & if \quad min_{l\notin vc(IWD)}(soil(i,l)) \geq 0 \\ soli(i,j) & if \quad -min_{l\notin vc(IWD)}(soil(i,l)) \; else \end{cases} \qquad (۵)$$

که در آن $\mathrm{soil}(i,j)$ میزان خاک حمل شده توسط قطره بین دو گره i و j است. سپس گره بازدید شده جدید j به لیست $V_c(IWD)$ اضافه می‌شود.

مرحله ۲. برای هر قطره جابجاشونده بین دو گره i و j سرعت آن توسط رابطه (۶) بروزرسانی می‌شود به طوری‌که هرچه خاک بیشتری بین دو گره وجود داشته باشد مقدار کمتری به سرعت آن افزوده می‌شود.

$$V^{IWD}(t+1) = V^{IWD}(t) + \frac{a_v}{b_v + c_v.soil^2(i,j)} \qquad (۶)$$

که در آن $V^{IWD}(t+1)$ سرعت بروز شده IWD است. پارامترهای ثابت a_v و b_v و c_v برای بروزرسانی سرعت به ترتیب برابر با ۱، ۰.۱ و ۱ لحاظ می‌شوند.

مرحله ۳. میزان خاکی که IWD از مسیر گره i به گره j جمع‌آوری می‌کند با استفاده از رابطه (۷) تعیین می‌شود.

$$\Delta soli(i,j) = \frac{a_s}{b_s + C_s.time^2\big(i,j;V^{IWD}(t+1)\big)} \qquad (۷)$$

که در آن پارامترهای ثابت a_s و b_s و c_s برای بروزرسانی خاک به ترتیب برابر با ۱، ۰.۱ و ۱ لحاظ می‌شوند. تابع $time$ در الگوریتم اصلی برابر با $\frac{HUD(j)}{V^{IWD}(t+1)}$ است و $\mathrm{HUD(j)}$ عدم مطلوبیت اکتشافی است که برحسب مسئله داده شده مشخص می‌شود.

مرحله ۴. راه‌حل بهترین-تکرار T^{IB} از همه راه‌حل‌های T^{IWD} یافت شده توسط IWD ها با استفاده از رابطه (۸) تعیین می‌شود.

$$\Delta soli(i,j) = \frac{a_s}{b_s + C_s.time^2\big(i,j;V^{IWD}(t+1)\big)} \qquad (۸)$$

که در آن $\mathrm{q(.)}$ کیفیت راه حل است.

مرحله ۵. خاک موجود در مسیرهایی که بهترین راه‌حل بهترین-تکرار T^{IB} جاری را شکل داده‌اند توسط رابطه (۹) بروزرسانی می‌شود.

$$soil(i,j) = (1+\rho_{IWD}).soil(i,j) - \rho_{IWD}.\frac{1}{(N_{IB}-1)}.soil_{IB}^{IWD} \qquad (۹)$$

[11] Shah-Hosseini

[Downloaded from joc-isice.ir on 2026-11-14]
[DOR: 20.1001.1.20088345.1400.15.2.1.8]
[DOI: 10.52547/joc.15.2.81]

که در آن ρ_{IWD} پارامتر بروزرسانی سراسری، N_{IB} تعداد گره‌های بهترین مسیر در تکرار جاری و $soil_{IB}^{IWD}$ قطره‌ای است که بهترین مسیر را پیموده است و $soil(i,j)$ میزان خاک بین دو گره در بهترین مسیر جاری است.

مرحله ۶. بهترین راه‌حل مجموع T^{TB} توسط بهترین راه‌حل فعلی T^{IB} با استفاده از رابطه (۱۰) بروزرسانی می‌شود.

$$T^{TB} = \begin{cases} T^{TB} & if\ q(T^{TB}) \geq q(T^{IB}) \\ T^{IB} & otherwise \end{cases} \qquad (۱۰)$$

مرحله ۷. الگوریتم با بهترین راه‌حل مجموع متوقف می‌شود.

درنهایت ممکن است بهترین مسیر مجموع با توجه به بهترین مسیر در تکرار جاری تغییر کند [۴۴]، [۴۵].

۳-۳ الگوریتم ترکیبی PSO-IWD

یکی از چالش‌های موجود در الگوریتم‌های تکاملی مبتنی بر جمعیت، مسئله گیرافتادن در نقاط بهینه محلی است. معمولا برای حل این مسئله از ترکیب الگوریتم های تکاملی با الگوریتم‌های جستجوی محلی مانند K2 یا تپه‌نوردی استفاده می‌شود. مسئله بعدی سرعت همگرایی ضعیف الگوریتم PSO است [۵]. در این مقاله، برای حل این دو چالش، یک روش فرااکتشافی ترکیبی جدید متشکل از الگوریتم‌های PSO و IWD پیشنهاد شده‌است. برای بهره‌گیری از مزیت تنوع جمعیت در PSO، ابتدا جمعیت اولیه با PSO تولید می‌شود. بروزرسانی بردار سرعت ذره i و موقعیت بعدی گره بترتیب توسط رابطه ۱ و ۲ متعلق به PSO انجام می‌شود. زمانی که مبدا و مقصد حرکت ذره در یک تکرار مشخص شدند الگوریتم IWD برای پیدا کردن بهترین مسیر از میان گره‌های میانی موجود بین این دو گره فراخوانی می‌شود. در واقع تلفیق الگوریتم‌های IWD و PSO منجر به اضافه شدن مولفه جهت به روند تکاملی می‌شود. مطمئنا در هر تکرار نیاز به ارزیابی توابع برازش هر دو الگوریتم می‌باشد. ضمن این که پارامتر W از مقادیر بزرگ شروع می‌شود تا فضای بزرگتری جستجو شود اما با افزایش تکرارها کاهش می‌یابد. شبه کد الگوریتم پیشنهادی در شکل ۲ ارائه شده است.

Input: N_p = Population Size
Output: The Scheduled Tasks List
For i=1 to N_p do (Generate the First Generation by PSO)
(1) Generate Particle P[i];
(2) Initialize V[i] & X[i] Randomly
(3) Compute Fitness Function for P[i];
(4) PBest [i] = P[i];
End For
GBest = Best P[i];
For i=1 to N_p do (Update Velocity and Position by PSO)
$V_i(t+1) = wV_i(t) + C_1R_{1,i}(t)\left(P_i(t) - X_i(t)\right) + C_2R_{2,i}(t)\left(P_g(t) - X_i(t)\right)$
$X_i(t+1) = X_i(t) + V_i(t+1)$
N_i = Number of Particles between $X_i(t)$ and $X_i(t+1)$
For i=1 to N_i do (Find the Best Path between Current Position and Next Position by IWD)
$soil[X_i(t), X_i(t+1)] = (1+\rho_{IWD}).soil[X_i(t), X_i(t+1)] - \rho_{IWD}.\frac{1}{(N_i-1)}.soil_i^{IWD}$
Compute Fitness Function for Soil[i];
Evaluate Soil [i];
If new Path is better
SBest [i] = Soil [i];
End For
Evaluate P[i];
If new Position is better
PBest [i] = P[i];
GBest = Best Particle Found
End For

شکل ۲: شبه کد الگوریتم ترکیبی PSO-IWD

[Downloaded from joc-isice.ir on 2026-11-14]
[DOR: 20.1001.1.20088345.1400.15.2.1.8]
[DOI: 10.52547/joc.15.2.81]

۴- پیاده‌سازی

در مرحله پیش‌آزمون (یعنی قبل از اِعمال در مسئله زمان‌بندی در محیط ابر) روش پیشنهادی روی برخی از توابع معیار اجرا خواهد شد. در این بخش چهار تابع معیار شناخته شده در حوزه بهینه‌سازی در نظر گرفته شده‌اند: توابع اکلی، روزن‌بروک، راستریگین و گری‌وانگ [۴۶]. جزئیات توابع معیار در جدول ۳ ارائه شده‌است. در تعریف توابع معیار، d به ابعاد مسئله اشاره دارد. پارامترهای اولیه اجرای الگوریتم روی توابع معیار نیز در جدول ۴ آورده شده‌است. کد برنامه در نرم‌افزار متلب R2016b v9.1 x.64 نوشته شد و برنامه روی یک سیستم خانگی با پردازنده Intel® Core™2 Duo E4500 و ۲ گیگابایت حافظه RAM اجرا شد. کد متلب توابع معیار از [۴۶] استخراج شده‌است.

جدول ۳: توابع معیار در نظر گرفته شده برای ارزیابی الگوریتم پیشنهادی

نام تابع	معادله	کران	مقدار کمینه جهانی
اکلی	$f(X)=-aexp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right)-exp\left(\frac{1}{d}\sum_{i=1}^{d}cos(cx_i)\right)+a+exp(1)$	$(-10,10)^d$	$f(X)=0, at\ X=(0,\dots,0)$
روزن‌بروک	$f(X)=\sum_{i=1}^{d-1}\left[100\left(x_{i+1}-x_i^2\right)^2+(x_i-1)^2\right]$	$(-10,10)^d$	$f(X)=0, at\ X=(1,\dots,1)$
راستریگین	$f(X)=10d+\sum_{i=1}^{d}[x_i^2-10cos(2\pi x_i)]$	$(-5,5)^d$	$f(X)=0, at\ X=(0,\dots,0)$
گری‌وانگ	$f(X)=\sum_{i=1}^{d}\frac{x_i^2}{4000}-\prod_{i=1}^{d}cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$(-10,10)^d$	$f(X)=0, at\ X=(0,\dots,0)$

جدول ۴: پارامترهای اولیه اجرای الگوریتم روی توابع معیار

اندازه جمعیت اولیه	حداکثر تعداد نسل (تکرار)	ابعاد مسئله (d)	تعداد اجرای تست روی هر تابع	شرایط توقف	پارامترهای ثابت
۵۰	۵۰	۲	۱۰	حداکثر تعداد نسل=۵۰ یا تغییر مقدار برازش کمتر از ۰.۰۰۱ در دو نسل متوالی	$a=20$ $b=0.2$ $c=2\pi$

معیارهای تعداد نسل تا همگرایی به جواب بهینه با عنوان «سرعت همگرایی» و زمان اجرای الگوریتم با عنوان «زمان CPU» برای مقایسه سه الگوریتم PSO-IWD، PSO، IWD در نظر گرفته شده‌اند. به ترتیب نتایج بهترین، متوسط، بدترین و انحراف معیار سرعت همگرایی در جدول ۵ و نتایج بهترین، متوسط، بدترین و انحراف معیار زمان CPU در جدول ۶ ذکر شده‌اند. مقدار متوسط برای سرعت همگرایی به حد پایین گرد شده‌است. همچنین تمام مقادیر زمان CPU نیز به سه رقم اعشار گرد شده‌اند.

همانطور که از نتایج جدول ۵ مشاهده می‌شود الگوریتم PSO-IWD از نظر مقدار بهترین در همه توابع بجز روزن‌بروک نسبت به دو روش دیگر عملکرد بهتر یا برابر دارد. همچنین از نظر مقدار متوسط و مقدار بدترین نیز بجز تابع روزن‌بروک از دو روش دیگر عملکرد بهتری دارد.

جدول ۵: مقایسه نتایج سرعت همگرایی الگوریتم‌های PSO-IWD، PSO، IWD روی توابع معیار

تابع معیار	الگوریتم	بهترین	متوسط	بدترین	انحراف معیار
اکلی	IWD	۲۸	۲۹	۳۲	۱.۸۲
	PSO	۲۷	۲۹	۳۱	۱.۶۳
	PSO-IWD	**۲۶**	**۲۸**	**۲۸**	۱.۱۵
روزن‌بروک	IWD	۴۱	۴۴	۴۹	۳.۳۱
	PSO	**۳۰**	**۳۰**	**۳۳**	۱.۷۳
	PSO-IWD	۴۳	۴۸	۴۸	۲.۸۸
راستریگین	IWD	۳۱	۳۳	۳۷	۲.۵۴
	PSO	۳۰	۳۱	۳۵	۲.۳۸
	PSO-IWD	**۳۰**	**۳۰**	**۳۴**	۲.۳۰
گری‌وانگ	IWD	۴۰	۴۲	۴۴	۱.۳۹
	PSO	۴۰	۴۱	۴۳	۱.۲۷
	PSO-IWD	**۳۴**	**۳۵**	**۳۸**	۱.۸۱

[Downloaded from joc-isice.ir on 2026-11-14]
[DOR: 20.1001.1.20088345.1400.15.2.1.8]
[DOI: 10.52547/joc.15.2.81]

جدول ۶: مقایسه نتایج زمان CPU (در واحد ثانیه) الگوریتم‌های PSO-IWD،

تابع معیار	الگوریتم	بهترین	متوسط	بدترین	انحراف معیار
اکلی	IWD	۰/۵۶۴	۰/۵۶۵	۰/۵۷۰	۰.۳۱۰
	PSO	۰/۵۷۱	۰/۵۷۶	۰/۵۷۸	۰.۳۱۰
	PSO-IWD	**۰/۵۶۳**	**۰/۵۶۳**	**۰/۵٦٦**	۰.۱۷۳
روزنبروک	IWD	۰/۹۱۶	۰/۹۲۶	۰/۹۴۰	۰.۹۹۳
	PSO	**۰/۸۱۸**	**۰/۸۲٦**	**۰/۸۲۸**	۰.۵۱۶
	PSO-IWD	۰/۸۶۱	۰/۸۷۰	۰/۹۱۸	۱.۰۸۱
راستریگین	IWD	۰/۶۶۰	۰/۶۷۲	۰/۶۷۰	۰.۷۲۰
	PSO	۰/۶۶۷	۰/۶۶۸	۰/۶۷۶	۰.۴۶۵
	PSO-IWD	**۰/٦٥١**	**۰/٦٥٩**	**۰/٦٦٥**	۰.۵۷۷
گری‌وانک	IWD	۰/۲۶۲	۰/۲۸۲	۰/۳۲۰	۰.۵۴۱
	PSO	۰/۲۶۰	۰/۲۹۴	۰/۳۲۶	۰.۲۴۷
	PSO-IWD	**۰/۲۱٦**	**۰/۲۳٦**	**۰/۲٤۲**	۰.۱۹۷

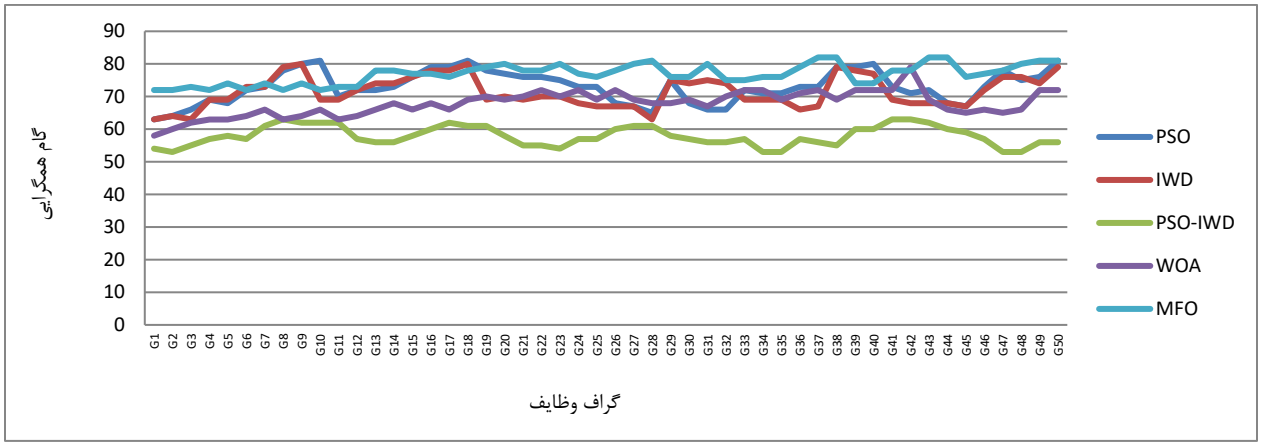
PSO، IWD روی توابع معیار

همانطور که از نتایج جدول ۶ مشاهده می‌شود الگوریتم PSO-IWD در همه توابع بجز روزنبروک از دو روش دیگر عملکرد بهتری دارد. در تابع روزنبروک بهترین عملکرد متعلق به PSO است و در تابع راستریگین نیز عملکرد نزدیکی نسبت به PSO-IWD دارد.

حال به مسئله اصلی باز می‌گردیم. در مسئله تحت بررسی، منابع ناهمگن در نظر گرفته شده‌اند یعنی قابلیت آن‌ها برای پردازش وظایف مختلف متفاوت

۵- نتایج و بحث

مقایسه سرعت همگرایی بین الگوریتم ترکیبی PSO-IWD با الگوریتم‌های PSO، IWD، WOA و MFO بصورت جداگانه روی ۵۰



شکل ۳: نتایج سرعت همگرایی حاصل از اجرای الگوریتم‌ها روی ۵۰ گراف با ۲۰ وظیفه

منظور از گام همگرایی عدد تکراری است که الگوریتم در آن به جواب بهینه همگرا شده‌است. زمانی که تعداد وظایف برابر با ۲۰ است گام همگرایی در الگوریتم PSO در بازه (۶۳ – ۸۱)، در الگوریتم IWD در بازه (۶۳ – ۸۰)، در الگوریتم WOA در بازه (۵۸ – ۷۲)، در الگوریتم

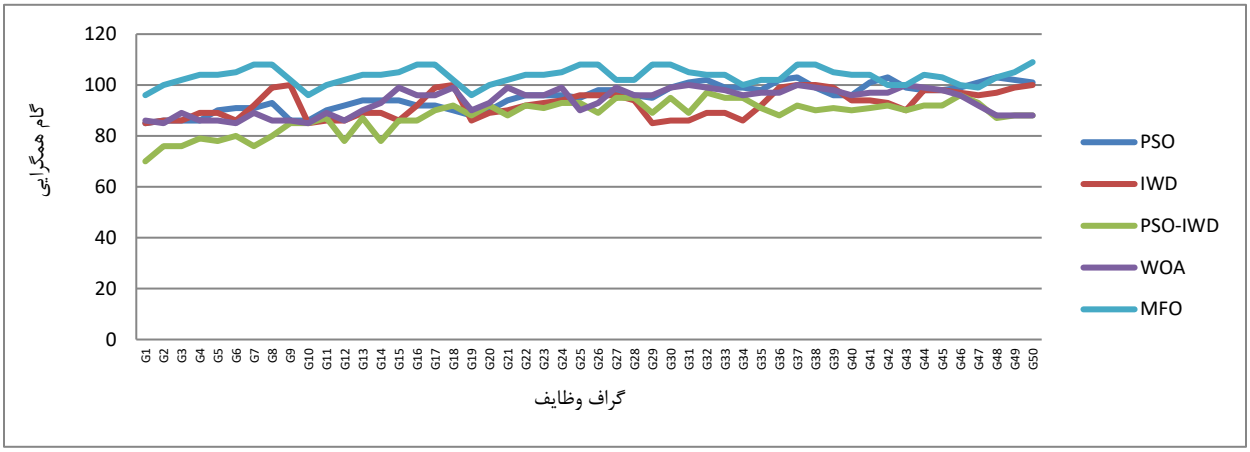
است و پردازش یک وظیفه روی منابع مختلف می‌تواند به زمان‌های اجرای متفاوت منجر شود. هزینه ارتباط بین وظایف نیز در نظر گرفته شده‌است. هزینه ارتباط یا هزینه لبه در گراف متناظر یعنی زمانی که برای انتقال نتیجه یک وظیفه به یک وظیفه مجاور صرف می‌شود. برای تولید هر DAG تصادفی به دو گراف $\mathbf{G1_{n\times n}}$ برای هزینه ارتباطی بین وظایف و $\mathbf{G2_{n\times m}}$ برای هزینه محاسباتی هر وظیفه نیاز است که در آن n تعداد وظایف و m تعداد منابع موجود است. هزینه محاسباتی بطور تصادفی بین ۵ تا ۲۰ و هزینه ارتباطی بطور تصادفی بین ۱ تا ۱۰ در واحد میلی‌ثانیه تولید شده‌است. پارامترهای الگوریتم در آزمایش‌ها به این صورت تنظیم شدند: اندازه جمعیت اولیه: ۵۰، حداکثر تعداد نسل‌ها: ۱۲۰، وزن اینرسی: ۰.۷، C_1 و C_2 به ترتیب ۲ و ۲. فاز تست در سه مرحله و هر بار با ۵۰ گراف با تعداد متفاوتی از وظایف (۲۰، ۴۰ و ۶۰) انجام شد.

نتایج حاصل از روش پیشنهادی علاوه بر الگوریتم‌های پایه PSO و IWD با نتایج حاصل از دو الگوریتم شناخته شده بهینه‌سازی وال (WOA) [۴۷] و بهینه‌سازی شعله/پروانه (MFO) [۴۸] که به ترتیب در سال های ۲۰۱۶ و ۲۰۱۵ توسعه یافته‌اند مقایسه خواهند شد. شرایط تست که در بالا ذکر شد برای همه پنج الگوریتم یکسان در نظر گرفته شده‌است. ارائه جزئیات بیشتر در مورد الگوریتم‌های مزبور بدلیل محدودیت فضا امکان‌پذیر نیست.

گراف با ۲۰ وظیفه، ۵۰ گراف با ۴۰ وظیفه و ۵۰ گراف با ۶۰ وظیفه به ترتیب در شکل‌های ۳، ۴ و ۵ نشان داده شده‌است. پارامترهای همه الگوریتم‌ها بطور یکسان مقداردهی شده‌اند. همچنین حداکثر تعداد تکرارها در هر ۵ الگوریتم برابر با ۱۲۰ تعیین شده‌است.

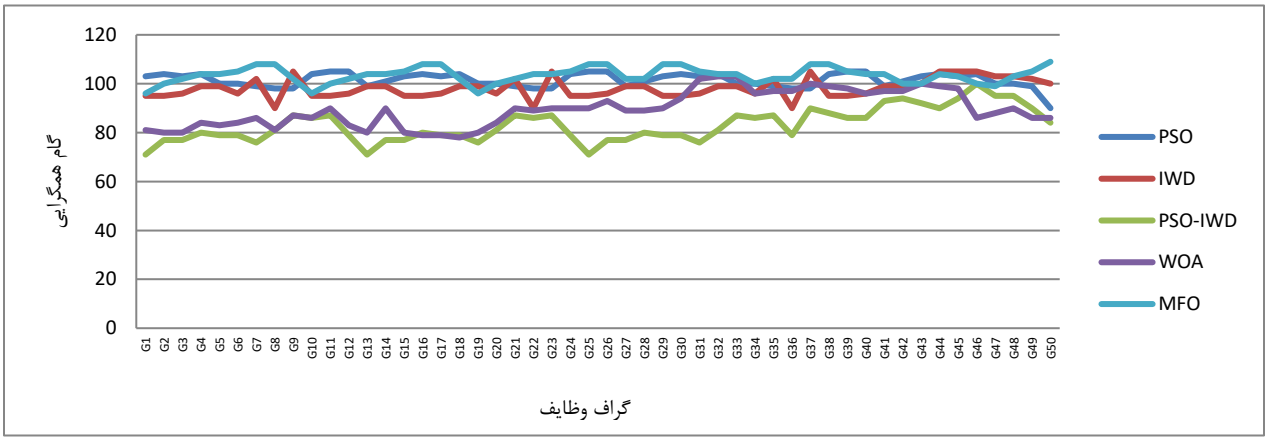
[Downloaded from joc-isice.ir on 2026-11-14]
[DOR: 20.1001.1.20088345.1400.15.2.1.8]
[DOI: 10.52547/joc.15.2.81]

MFO در بازه (۷۲ – ۸۲) و در الگوریتم ترکیبی PSO-IWD در بازه (۵۳ – ۶۳) تغییر می‌کند. نزدیکی نتایج دو الگوریتم PSO و IWD در شکل کاملا مشهود است. ضمن این که MFO بطور متوسط بدترین نتایج را داشته است. برتری الگوریتم ترکیبی PSO-IWD نیز در تمام طول تکرارها (بجز یک مورد در گراف ۹) قابل توجه است.



شکل ۴: نتایج سرعت همگرایی حاصل از اجرای الگوریتم‌ها روی ۵۰ گراف با ۴۰ وظیفه

به همین ترتیب زمانی که تعداد وظایف برابر با ۴۰ است گام همگرایی در الگوریتم PSO در بازه (۸۵ – ۱۰۳)، در الگوریتم IWD در بازه (۸۵ – ۱۰۰)، در الگوریتم WOA در بازه (۸۵ – ۱۰۰)، در الگوریتم MFO در بازه (۹۶ – ۱۰۸) و در الگوریتم ترکیبی PSO-IWD در بازه (۷۷ – ۹۷) تغییر می‌کند. مجددا همانطور که در شکل مشهود است MFO بدترین نتایج را در این سناریو کسب کرده است. در بین سه سناریو نتایج الگوریتم‌ها در سناریوی دوم نسبت به دو سناریوی دیگر نزدیک‌تر است. نتایج IWD و WOA به هم نزدیک و در ۱۶ مورد کاملا یکسان است. الگوریتم IWD در ۲۱ درصد، الگوریتم PSO-IWD در ۷۲ درصد و الگوریتم WOA در ۷ درصد موارد بهترین نتیجه را کسب کرده‌اند.



شکل ۵: نتایج سرعت همگرایی حاصل از اجرای الگوریتم‌ها روی ۵۰ گراف با ۶۰ وظیفه

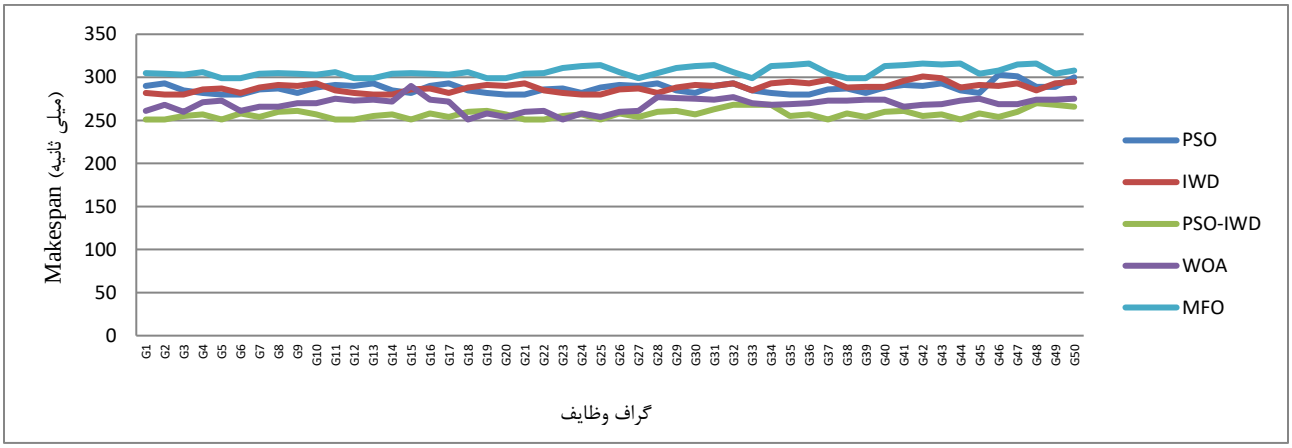
به همین ترتیب زمانی که تعداد وظایف برابر با ۶۰ است، گام همگرایی در الگوریتم PSO در بازه (۹۰ – ۱۰۵)، در الگوریتم IWD در بازه (۹۵ – ۱۰۵)، در الگوریتم WOA در بازه (۸۱ – ۱۰۱)، در الگوریتم MFO در بازه (۹۶ – ۱۰۹) و در الگوریتم ترکیبی PSO-IWD در بازه (۷۱ – ۱۰۲) تغییر می‌کند. هر چه تعداد وظایف بیشتر می‌شود نتایج WOA به PSO-IWD و نتایج PSO به IWD نزدیک‌تر می‌شود. در سناریوی سوم هم مشابه سناریوهای قبلی MFO بطور متوسط بدترین نتایج را کسب کرده‌است. در ۲۶ درصد از موارد WOA و در بقیه موارد PSO-IWD بهترین نتیجه را داشته‌اند. نتایج بهترین، متوسط، بدترین و انحراف معیار سرعت همگرایی در جدول ۷ ارائه شده‌اند. شکل‌های ۶، ۷ و ۸ به ترتیب

[Downloaded from joc-isice.ir on 2026-11-14]
[DOR: 20.1001.1.20088345.1400.15.2.1.8]
[DOI: 10.52547/joc.15.2.81]

Make-Span حاصل از اعمال الگوریتم‌های PSO-IWD، PSO، IWD، WOA و MFO روی ۵۰ گراف با تعداد وظایف مختلف را نشان می‌دهند. پارامترهای همه الگوریتم‌ها بطور یکسان مقداردهی شده‌اند. همچنین حداکثر تعداد تکرارها در هر ۳ سناریو برابر با ۱۲۰ تعیین شده‌است. مقادیر Make-Span پس از ۱۲۰ تکرار (معیار توقف الگوریتم) ثبت شده‌اند صرف نظر از این که الگوریتم به همگرایی رسیده است یا خیر.

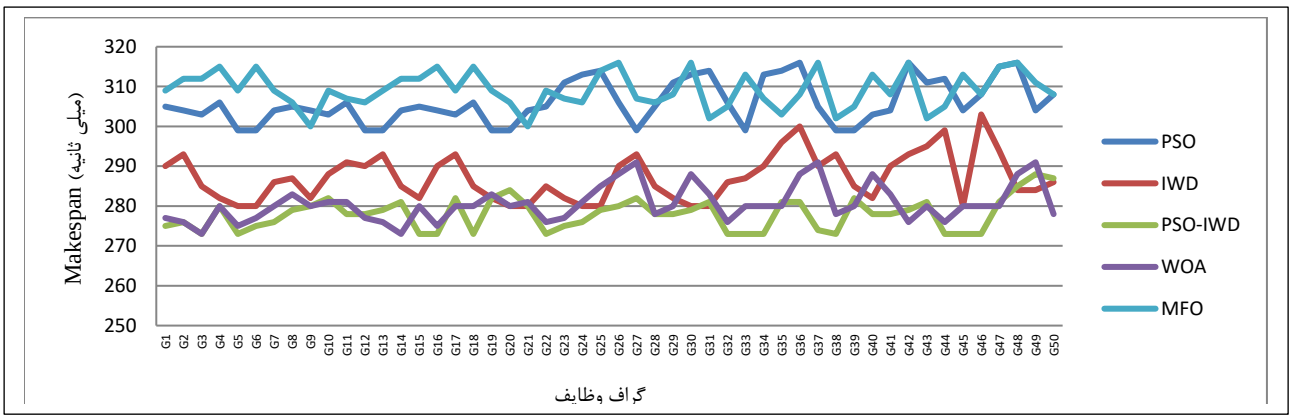
جدول ۷. نتایج بهترین، متوسط، بدترین و انحراف معیار سرعت همگرایی.

سناریوی آزمون	الگوریتم	بهترین	متوسط	بدترین	انحراف معیار
سناریوی اول	IWD	۶۳	۷۴	۸۰	۷.۲۳
	PSO	۶۳	۷۸	۸۱	۶.۵۸
	PSO-IWD	**۵۳**	**۵۷**	**۶۳**	۴.۰۸
	WOA	۵۸	۶۸	۷۲	۶.۲۱
	MFO	۷۲	۸۰	۸۲	۴.۷۶
سناریوی دوم	IWD	۸۵	۹۴	۱۰۰	۶.۲۴
	PSO	۸۵	۹۸	۱۰۳	۸.۰۴
	PSO-IWD	**۷۰**	**۸۸**	**۹۷**	۱۱.۶۱
	WOA	۸۵	۹۴	۱۰۰	۶.۲۴
	MFO	۹۶	۱۰۴	۱۰۸	۵.۱۶
سناریوی سوم	IWD	۹۵	۱۰۳	۱۰۵	۴.۷۶
	PSO	۹۰	۹۶	۱۰۵	۶.۲۴
	PSO-IWD	**۷۱**	**۸۴**	۱۰۲	۱۱.۹۰
	WOA	۸۱	۹۹	**۱۰۱**	۱۰.۶۴
	MFO	۹۶	۱۰۴	۱۰۹	۵.۴۴



شکل ۶: نتایج Make-Span حاصل از اجرای الگوریتم‌ها روی ۵۰ گراف با ۲۰ وظیفه

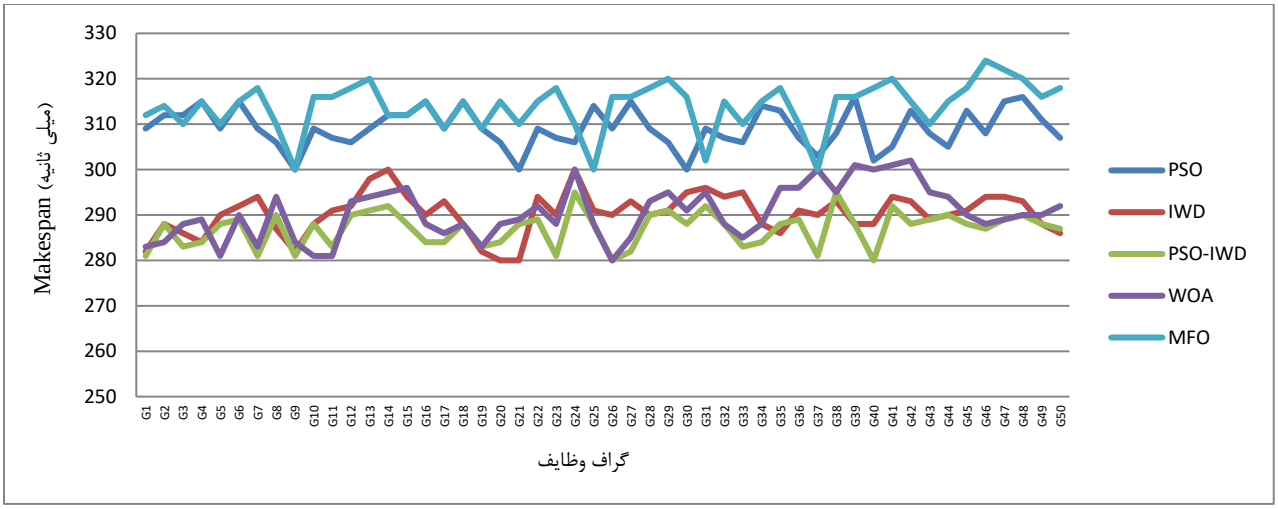
همانطور که در شکل مشهود است، در سناریوی اول نتایج PSO و IWD خیلی نزدیک به هم هستند. بدترین عملکرد مربوط به الگوریتم MFO همچنین PSO-IWD در ۸۶ درصد موارد بهترین عملکرد را داشته است.



شکل ۷: نتایج Make-Spanحاصل از اجرای الگوریتم ها روی ۵۰ گراف با ۴۰ وظیفه

[Downloaded from joc-isice.ir on 2026-11-14]
[DOR: 20.1001.1.20088345.1400.15.2.1.8]
[DOI: 10.52547/joc.15.2.81]

در سناریوی دوم PSO بیشترین تنزل را داشته است و حتی در ۱۳ مورد از MFO بدتر است. مشابه با سناریوی اول بهترین عملکرد مربوط به الگوریتم PSO-IWD است گر چه در ۱۰ مورد از WOA عملکرد بدتری داشته است.



شکل ۸: نتایج Make-span حاصل از اجرای الگوریتم‌ها روی ۵۰ گراف با ۶۰ وظیفه

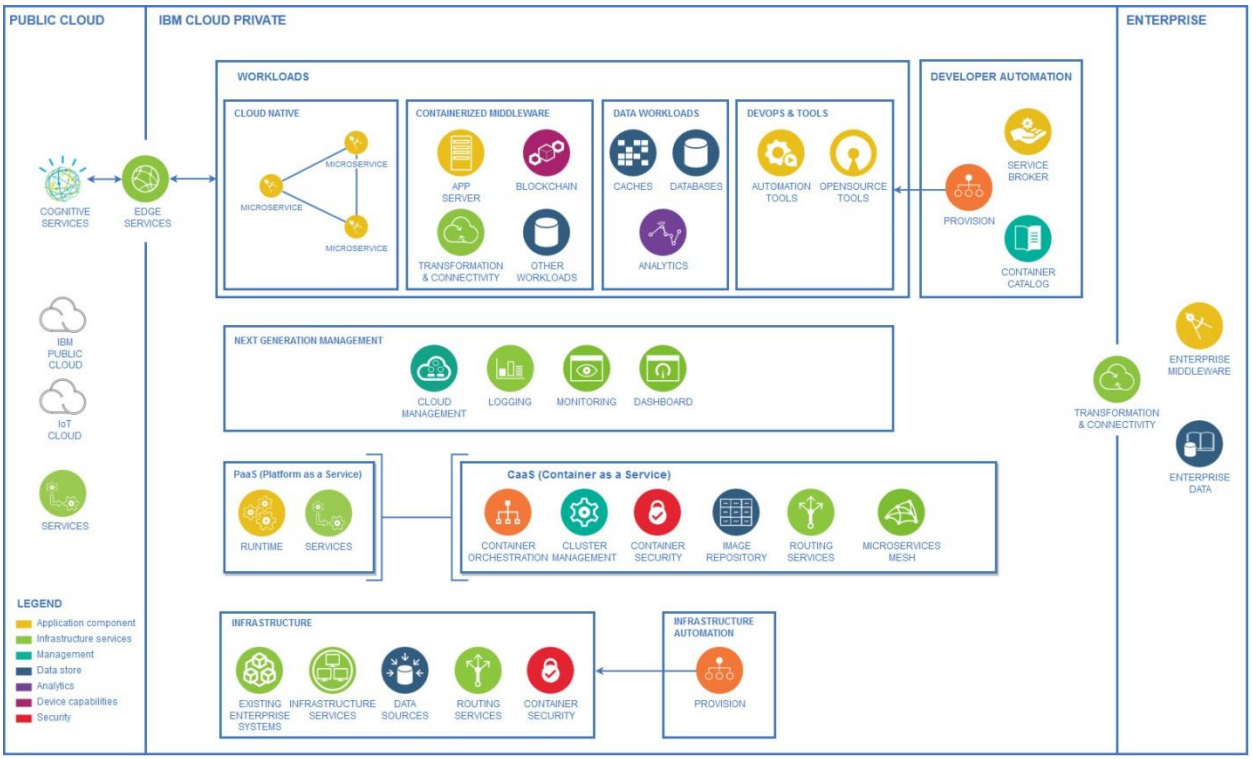
در سناریوی سوم نتایج IWD به دو الگوریتم PSO-IWD و WOA نزدیک‌تر شده‌است و حتی در ۵ مورد بهترین عملکرد را داشته است. اما همچنان PSO-IWD در ۶۷ درصد از موارد بهترین عملکرد را به خود اختصاص داده است. همانطور که در شکل‌های ۶، ۷ و ۸ نشان داده شده است روند کلی بدین صورت است که با افزایش تعداد وظایف، Make-Span افزایش می‌یابد، عملکرد PSO تنزل و عملکرد IWD تقویت می‌شود. نتایج بهترین، متوسط، بدترین و انحراف معیار Make-Span در جدول ۸ ارائه شده‌اند.

جدول ۸: نتایج بهترین، متوسط، بدترین و انحراف معیار Make-Span (در واحد میلی‌ثانیه)

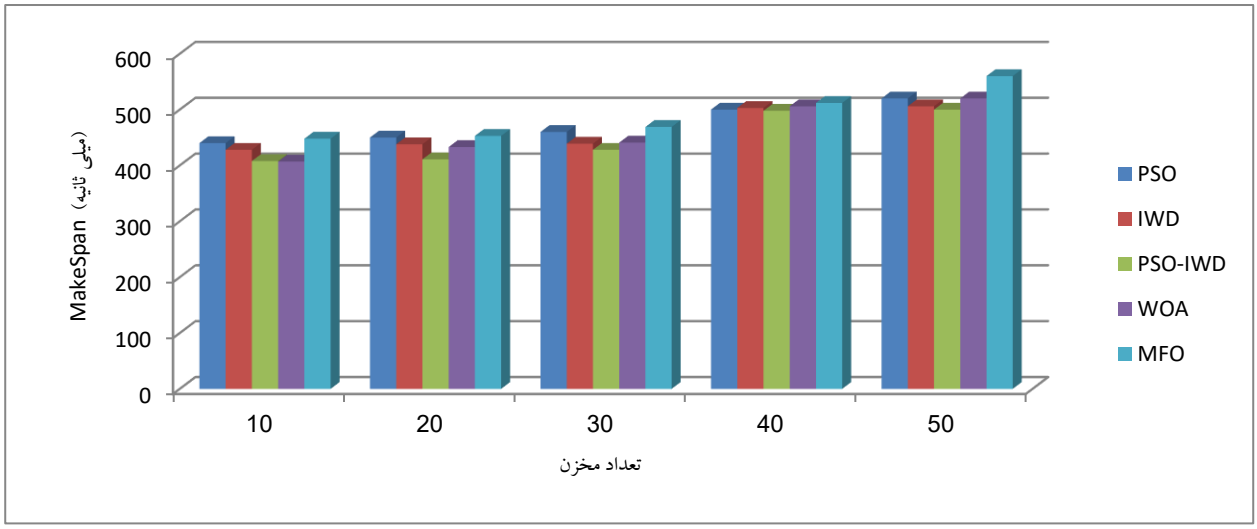
سناریوی آزمون	الگوریتم	بهترین	متوسط	بدترین	انحراف معیار
سناریوی اول	IWD	۲۸۰	۲۹۶	۳۰۱	۹.۸۶
	PSO	۲۸۰	۲۹۸	۳۰۳	۱۰.۷۸
	PSO-IWD	**۲۵۱**	**۲۵۶**	**۲۷۰**	۸.۵۹
	WOA	۲۶۰	۲۸۱	۲۹۰	۸.۲۲
	MFO	۲۹۹	۳۱۰	۳۱۶	۷.۲۵
سناریوی دوم	IWD	۲۸۲	۳۰۲	۳۰۶	۸.۹۶
	PSO	۲۹۹	۳۰۹	۳۱۶	۷.۰۴
	PSO-IWD	**۲۷۳**	**۲۸۰**	**۲۸۸**	۶.۱۳
	WOA	۲۷۲	۲۷۸	۲۹۱	۳.۸۷
	MFO	۳۰۶	۳۱۳	۳۱۶	۴.۳۹
سناریوی سوم	IWD	۲۸۲	۲۹۶	۳۰۰	۸.۴۰
	PSO	۳۰۰	۳۱۲	۳۱۶	۷.۳۰
	PSO-IWD	**۲۸۱**	**۲۸۸**	**۲۹۵**	۵.۷۱
	WOA	۲۸۱	۲۸۹	۳۰۱	۴۱.۱۲
	MFO	۳۰۰	۳۱۹	۳۲۴	۱۱.۳۴

آخرین مجموعه آزمایش‌ها مربوط به اِعمال الگوریتم پیشنهادی روی جریان‌های کاری تجاری در یک ابر مصنوعی است. ساختار سازمان-ابر که در شکل ۹ نشان داده شده است در نرم افزار Windows Azure Workflows برای این منظور توسعه یافته‌است. ابر پیشنهادی یک ابرِ «پلت‌فرم در قالب سرویس یا PaaS است که در چهار لایه ایجاد شده‌است شامل بارکاری، مدیریت تولید، PaaS و زیرساخت. این ساختار چهار لایه‌ای ساده‌ترین حالت ممکن برای تعریف یک ابر خصوصی است و لایه‌های مزبور تا ۱۲ مورد قابل افزایش هستند. منابع از سمت ابر خصوصی در قالب مخزن‌ها به سازمان تخصیص می‌یابند. تعداد مخزن‌ها و نوع ارتباط آن‌ها از طریق ابزار مدیریت خوشه قابل‌پیکربندی است. شکل ۱۰ نتایج مربوط به Make-Span حاصل از الگوریتم‌های PSO-IWD، PSO، IWD، WOA و MFO در ۵ سناریو با ۱۰، ۲۰، ۳۰، ۴۰، ۵۰ مخزن را نشان می‌دهد.

[Downloaded from joc-isice.ir on 2026-11-14]
[DOR: 20.1001.1.20088345.1400.15.2.1.8]
[DOI: 10.52547/joc.15.2.81]



شکل ۹: ساختار ابر توسعه یافته در نرم افزار Windows Azure Workflows



شکل ۱۰: نتایج حاصل از الگوریتم‌های PSO-IWD، PSO، IWD، WOA و MFO در ۵ سناریو با ۱۰، ۲۰، ۳۰، ۴۰، ۵۰ مخزن

در سناریوی اول یعنی زمانی که تعداد مخازن برابر با ۱۰ است WOA بهترین نتیجه را کسب کرده‌است اما در سایر سناریوها PSO-IWD برتری محسوسی نسبت به الگوریتم‌های دیگر دارد. با رشد تعداد مخازن عملکرد WOA تنزل پیدا کرده‌است و MFO نیز در مجموع بدترین عملکرد را داشته‌است.

۱-۵ آزمون آماری فریدمن

آزمون فریدمن یک آزمون ناپارامتری است که برای مقایسه چندین روش که حداقل در سطح رتبه‌ای اندازه‌گیری می‌شوند، مورد استفاده قرار می‌گیرد. این آزمون می‌تواند در مورد داده‌های پیوسته (فاصله‌ای یا نسبی) نیز بکار رود، اما در هنگام محاسبه این داده‌ها نیز رتبه‌بندی آن‌ها مدنظر

[Downloaded from joc-isice.ir on 2026-11-14]
[DOR: 20.1001.1.20088345.1400.15.2.1.8]
[DOI: 10.52547/joc.15.2.81]

قرار می‌گیرد. در این آزمون برای اجرای تحلیل واریانس داده‌های تکرار شده فرضیاتی مانند نرمال بودن توزیع، برابری واریانس‌ها و پیوسته بودن مقیاس مطرح نیستند. فرضیه صفر در آزمون فریدمن به این صورت بیان می‌شود که هیچ اختلاف معناداری بین روش‌ها وجود ندارد. آماره فریدمن از رابطه (۱۱) محاسبه می‌شود [۴۹]. که در آن K تعداد روش‌های تحت مقایسه است، N تعداد مجموعه داده‌ها (سناریوهای آزمون) است و R میانگین رتبه یک روش در میان تمام روش‌ها در هر آزمون می‌باشد. در مورد مسئله تحت بررسی K=5، N=5 و مقدار بحرانی طبق جدول فریدمن برای K=5 و N=5 و سطح معنی‌داری ۵ درصد برابر با ۷.۸۰۰ می باشد. اگر مقدار آماره فریدمن از مقدار بحرانی بیشتر باشد فرضیه صفر رد می‌شود. در مورد نتایج شکل ۱۰ میانگین رتبه‌ها برای پنج روش MFO، WOA، PSO، IWD و PSO-IWD به ترتیب در ۵ سناریو برابر با ۴.۸، ۲.۶، ۳.۴، ۲.۶ و ۱.۲ هستند. طبق رابطه (۱۱) آماره فریدمن برابر است با ۸.۵۶. چون آماره فریدمن از مقدار بحرانی بیشتر است فرضیه ۰ رد می‌شود و فرضیه جایگزین پذیرفته می‌شود یعنی اختلاف بین روش‌ها معنادار است.

۶- نتیجه‌گیری و کارهای آینده

هدف از زمان‌بندی وظایف در سیستم‌های توزیع شده، تخصیص وظایف به منابع ناهمگن است بگونه‌ای که برخی از معیارهای عملکرد، مانند زمان اجرا یا توازی بهینه شوند. زمان‌بندی وظایف یک مسئله NP–کامل است، به همین دلیل از الگوریتم‌های اکتشافی و فرااکتشافی برای حل آن استفاده می‌شود. در این مقاله یک الگوریتم ترکیبی PSO-IWD برای زمان‌بندی وظایف در محیط‌هایی با منابع ناهمگن مانند ابر پیشنهاد شده است. نتایج حاصل از اِعمال این الگوریتم روی DAG های مصنوعی با اندازه‌های مختلف نشان می‌دهد که روش پیشنهادی از نظر سرعت همگرایی و زمان پردازش کل وظایف نسبت به الگوریتم‌های PSO و IWD به‌صورت جداگانه و الگوریتم‌های MFO، WOA موثرتر است. همچنین نتایج حاصل از اِعمال الگوریتم پیشنهادی روی جریان‌های کاری ابر مصنوعی تولید شده در نرم افزار AZURE حاکی از برتری الگوریتم ترکیبی PSO-IWD در مقایسه با روش‌های دیگر دارد. رویکرد در نظر گرفته شده در این مقاله نسبت به مسئله زمان‌بندی وظایف یک رویکرد تک‌هدفه (زمان پردازش کل وظایف) بود اگر چه شاخص سرعت همگرایی الگوریتم هم مورد توجه قرار گرفت. در کارهای آینده ما قصد داریم الگوریتم پیشنهادی را در قالب رویکردهای چندهدفه با در نظر گرفتن معیارهایی مانند میزان استفاده از پردازنده، میزان حرارت تولید شده و همچنین در نظر گرفتن مولفه‌هایی از قبیل زمان‌بندی بلادرنگ و توازن بار پویا تعمیم دهیم. ضمن این که آزمون الگوریتم ترکیبی PSO-IWD روی گراف‌های جهان واقعی و توسعه نسخه کوانتومی الگوریتم IWD بعنوان کارهای تحقیقاتی آینده مطلوب خواهند بود.

مراجع

[1] Erl Thomas, Puttini Ricardo, Mahmood Zaigham, Cloud Computing: Concepts, Technology & Architecture (1st Edition), Prentice Hall, 2013.

[2] Rountree Derrick, Castrillo Ileana, Understanding the Fundamentals of Cloud Computing in Theory and Practice (1st Edition), Syngress, 2013.

[3] Kumar Sehgal, Naresh, Cloud Computing: Concepts and Practices, Springer, 2018.

[4] Ullman J.D, 1975, "NP-complete scheduling problems", Journal of Computer and System Sciences, 10(3), pp. 384–393.

[5] Lee K.Y, Park J.B, 2006, "Application of Particle Swarm Optimization to Economic Dispatch Problem: Advantages and Disadvantages", 2006 IEEE PES Power Systems Conference and Exposition, Atlanta, GA, USA. pp. 188–192.

[6] Talbi El-Ghazali, Metaheuristics: From Design to Impelementation, John Wiley and sons, 2009.

[7] Arnav W, Theng D, 2016, "A survey on different scheduling algorithms in cloud computing", In 2016 2nd International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), pp. 665-669.

[8] Santhosh B, Manjaiah D. H, Suresh L.P, 2016, "A survey of various scheduling algorithms in cloud environment", 2016 International Conference on Emerging Technological Trends (ICETT), Kollam, India.

[9] SudhirShenai V, 2012, "Survey on Scheduling Issues in Cloud Computing", Procedia Engineering, 38, pp. 2881-2888.

[10] Muhuri P.K, Rauniyar A, Nath R, 2019, "On arrival scheduling of real-time precedence constrained tasks on multi-processor systems using genetic algorithm", Future Generation Computer Systems, 93, pp. 702-726.

[11] Sharma M, Garg R, 2019, "HIGA: Harmony-inspired genetic algorithm for rack-aware energy-efficient task scheduling in cloud data centers", Engineering Science and Technology, an International Journal, 23, pp. 211-224.

[12] Alharkan I, Saleh M, Ghaleb M.A, Kaid H, Farhan A, Almarfadi A, 2019, "Tabu search and particle swarm optimization algorithms for two identical parallel machines scheduling problem with a single

[Downloaded from joc-isice.ir on 2026-11-14]
[DOR: 20.1001.1.20088345.1400.15.2.1.8]
[DOI: 10.52547/joc.15.2.81]

[23] Thanka M.R, Maheswari P.U, Edwin E.B, 2017, "An improved efficient: Artificial Bee Colony algorithm for security and QoS aware scheduling in cloud computing environment", Cluster Computing, 20, pp. 1-9.

[24] Reddy GN, Kumar S.P, 2017, "Multi Objective Task Scheduling Algorithm for Cloud Computing Using Whale Optimization Technique", Smart and Innovative Trends in Next Generation Computing Technologies, pp. 286-297.

[25] Jiang T, Zhang C, Sun QM, 2017, "Green Job Shop Scheduling Problem With Discrete Whale Optimization Algorithm", IEEE Access, 7, pp. 43153 – 43166.

[26] Wu S.Y, Zhang P, Li F, Gu F, Pan Y.I, 2017, "A hybrid discrete particle swarm optimization-genetic algorithm for multi-task scheduling problem in service oriented manufacturing systems", Journal of Central South University, 23, pp. 421-429.

[27] Abdelaziz F.B, Mir H, 2016, "An Optimization Model and Tabu Search Heuristic for Scheduling of Tasks on a Radar Sensor", IEEE Sensors Journal, 16, pp. 6694-6702.

[28] Wu S.Y, Zhang P, Li F, Gu F, Pan Y, 2016, "A hybrid discrete particle swarm optimization-genetic algorithm for multi-task scheduling problem in service oriented manufacturing systems", Journal of Central South University, 23, pp. 421-429.

[29] Akbari M, Rashidi H, "A multi-objectives scheduling algorithm based on cuckoo optimization for task allocation problem at compile time in heterogeneous systems", Expert Systems with Applications, 60, pp. 234-248.

[30] Jiang Y.S, Chen W.M, 2015, "Task scheduling for grid computing systems using a genetic algorithm", The Journal of Supercomputing, 71, pp. 1357-1377.

[31] Ramezani F, Lu J, Hussain F.K, 2014, "Task-Based System Load Balancing in Cloud Computing Using Particle Swarm Optimization", International Journal of Parallel Programming, 42, pp. 739-754.

[32] Bukata L, Šůcha P, Hanzálek Z, 2015, "Solving the Resource Constrained Project Scheduling Problem using the parallel Tabu Search designed for the CUDA platform", Journal of Parallel and Distributed Computing, 77, pp. 58-68.

[33] Moschakis I.A, Karatza H.D, 2015, "Multi-criteria scheduling of Bag-of-Tasks applications on heterogeneous interlinked clouds with simulated annealing", Journal of Systems and Software, 101, pp. 1-14.

[34] Xu Y, Li K, Hu J, Li K, 2014, "A genetic algorithm for task scheduling on heterogeneous computing server", Journal of King Saud University - Engineering Sciences, xx, pp. xx-xx, (In-press).

[13] Adhikari M, Srirama S.N, 2019, "Multi-objective accelerated particle swarm optimization with a container-based scheduling for Internet-of-Things in cloud environment", Journal of Network and Computer Applications, 137, pp. 35-61.

[14] Mansouri N, HasaniZade B.H, Javidi M.M, 2019, "Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory", Computers & Industrial Engineering, 130, pp. 597-633.

[15] AbdElaziz M, Ewees A.A, AliIbrahim R, Lu S.F, 2019, "Opposition-based moth-flame optimization improved by differential evolution for feature selection", Mathematics and Computers in Simulation, 168, pp. 48-75.

[16] Casas I, Taheri J, Ranjan R, Wang L, Zomaya A.Y, 2018, "GA-ETI: An enhanced genetic algorithm for the scheduling of scientific workflows in cloud environments", Journal of Computational Science, 26, pp. 318-331.

[17] Wu G, Wang H, Pedrycz W, Li H, Wang L, 2017, "Satellite observation scheduling with a novel adaptive simulated annealing algorithm and a dynamic task clustering strategy", Computers & Industrial Engineering, 113, pp. 576-588.

[18] Baizid K, Yousnadj A, Meddahi A, Chellali R, Iqbal J, 2015, "Time scheduling and optimization of industrial robotized tasks based on genetic algorithms", Robotics and Computer-Integrated Manufacturing, 34, pp. 140-150.

[19] Kamalinia A, Ghaffari A, 2017, "Hybrid Task Scheduling Method for Cloud Computing by Genetic and DE Algorithms", Wireless Personal Communications, 97, pp. 6301-6323.

[20] Keshanchi B, Souri A, Navimipour N.J, 2017, "An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: Formal verification, simulation, and statistical testing", Journal of Systems and Software, 124, pp. 1-21.

[21] Boveiri HR, 2017 "An incremental ant colony optimization based approach to task assignment to processors for multiprocessor scheduling", Frontiers of Information Technology & Electronic Engineering, 18, pp. 498-510.

[22] Moon Y.J, Yu H.C, Gil J.M, Lim J.B, 2017, "A slave ants based ant colony optimization algorithm for task scheduling in cloud computing environments, Human-centric Computing and Information Sciences", 12, pp. 128-139.

[Downloaded from joc-isice.ir on 2026-11-14]
[DOR: 20.1001.1.20088345.1400.15.2.1.8]
[DOI: 10.52547/joc.15.2.81]

[42] Annicchiarico W.D, Cerolazza M, 2019, “Improved Dynamical Particle Swarm Optimization Method for Structural Dynamics”, Mathematical Problems in Engineering, 37, pp. 58-69.

[43] Kennedy James, Particle Swarm Optimization, Encyclopedia of Machine Learning. Springer, Boston, MA, 2011.

[44] Shah-Hosseini H, 2009, "The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm," International Journal of Bio-Inspired Computation, 1, pp. 71-79.

[45] Shah-Hosseini H, 2012, “An approach to continuous optimization by the Intelligent Water Drops algorithm”, Procedia Social and Behavioral Sciences, 32, pp. 224-229.

[46] Surjanovic S, Bingham D, 2013, “Virtual Library of Simulation Experiments: Test Functions and Datasets”, Retrieved August 31, 2019, from http://www.sfu.ca/~ssurjano.

[47] Mirjalili S, Lewis A, 2016, “The Whale Optimization Algorithm”, Advances in Engineering Software, 95, pp. 51-67.

[48] Mirjalili S, 2015, “Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm”, Knowledge-Based Systems, 89, pp. 228-249.

[49] Wittkowski K.M, (1988). "Friedman-Type statistics and consistent multiple comparisons for unbalanced designs with missing data". Journal of the American Statistical Association”, 83(404), pp. 1163–1170.

systems using multiple priority queues”, Information Sciences, 270, pp. 255-287.

[35] Samal A.K, Mall R, Tripathy C, 2014, “Fault tolerant scheduling of hard real-time tasks on multiprocessor system using a hybrid genetic algorithm”, Swarm and Evolutionary Computation, 14, pp. 92-105.

[36] Lu H, Liu J, Niu R, Zhu Z, 2014, “Fitness distance analysis for parallel genetic algorithm in the test task scheduling problem”, Soft Computing, 18, pp. 2385-2396.

[37] Davidović T, Šelmić M, Teodorović D, Ramljak D, 2012, “Bee colony optimization for scheduling independent tasks to identical processors”, Journal of Heuristics, 18, pp. 549-569.

[38] Aziza H, Krichen S, 2018, “Bi-objective decision support system for task-scheduling based on genetic algorithm in cloud computing”, Computing, 100, pp. 65-91.

[39] DerChou F, 2013 “Particle swarm optimization with cocktail decoding method for hybrid flow shop scheduling problems with multiprocessor tasks”, International Journal of Production Economics, 141, pp. 137-145.

[40] Orsila H, Salminen E, Hämäläinen T, 2013, “Recommendations for using Simulated Annealing in task mapping”, Design Automation for Embedded Systems, 17, pp. 53-85.

[41] Kennedy James, Eberhart Russell, Swarm Intelligence (1st ed), Academic Press, San Diego, CA, 2001.

[Downloaded from joc-isice.ir on 2026-11-14]
[DOR: 20.1001.1.20088345.1400.15.2.1.8]
[DOI: 10.52547/joc.15.2.81]