فصلنامه علمی مطالعات مدیریت صنعتی – سال هجدهم، شماره۵۸، پاییز ۹۹ (نوع مقاله: پژوهشی)
صفحات ۳۳۵-۳۰۷
DOI: 10.22054/jims.2020.47551.2394

الگوریتم‌های فراابتکاری برای مسئله زمان‌بندی جریان کارگاهی مونتاژ دومرحله‌ای با در نظر گرفتن زمان‌های آماده‌سازی ماشین‌ها

مهدی یزدانی*

تاریخ دریافت:۹۸/۸/۱۸- تاریخ پذیرش:۹۹/۱/۳۱

چکیده

در این مقاله، مساله زمان بندی خط تولید جریان کارگاهی مونتاژ دو مرحله‌ای با در نظر گرفتن زمان‌های آماده‌سازی مستقل از توالی و با هدف کمینه‌سازی مجموع زمان تکمیل کل سفارشات مطالعه می‌شود. در این مسئله چند سفارش برای یک نوع محصول وجود دارد. هر محصول سفارش‌داده‌شده از چند قطعه متفاوت تشکیل شده است. در ابتدا، قطعات در یک محیط جریان کارگاهی با وجود چند ماشین (ایستگاه) مختلف تولید می‌شوند و سپس در یک ایستگاه مونتاژ تبدیل به محصول نهایی می‌شوند. این مقاله سه الگوریتم فراابتکاری جستجوی همسایگی متغیر موازی، ایمنی مصنوعی و شبیه‌سازی تبرید را برای حل این مسئله ارائه داده است. روش طراحی آزمایشات تاگوچی جهت تنظیم پارامترها و عملگرهای الگوریتم‌های ارائه شده به کار گرفته شده است. همچنین با استفاده از آزمایش‌های عددی، عملکرد الگوریتم‌های پیشنهادی ارزیابی شده است. نتایج نشان می‌دهد الگوریتم جستجوی همسایگی متغیر موازی پیشنهاد شده از الگوریتم‌های دیگر برای حل این مساله بهتر عمل می‌کند.

کلمات کلیدی: مسئله جریان کارگاهی مونتاژ دو مرحله ای، زمان‌بندی، زمان‌های آماده‌سازی مستقل از توالی، الگوریتم فراابتکاری.

* استادیار، گروه مهندسی صنایع، دانشکده مهندسی صنایع و مکانیک، واحد قزوین، دانشگاه آزاد اسلامی، قزوین، ایران
(نویسنده مسئول) m_yazdani@qiau.ac.ir

مقدمه

زمان‌بندی کارها یکی از مهمترین وظایف سیستم‌های خدماتی و تولیدی است. پیدا کردن بهترین زمانبندی می‌تواند بسته به محیط کارگاه، محدودیت‌های فرایند و شاخص‌های عملکرد خیلی آسان یا خیلی سخت باشد (دمیر و کورسات، ۲۰۱۳). یکی از این مسائل، مساله جریان کارگاهی مونتاژ دو مرحله ای[1] (TSAFP) است که کاربردهای زیادی در صنایعی مانند کارخانه مونتاژ ماشین آتش‌نشانی (لی و همکاران، ۱۹۹۳)، تولید کامپیوتر شخصی (پاتس وهمکاران، ۱۹۹۵)، سیستم‌های پایگاه داده توزیع شده (اللهوردی و الانزی، ۲۰۰۶) و غیره دارد. این مساله در طول دو دهه‌ی اخیر توجه بیشتری را به خود جلب کرده است. بخش اول این مساله در اصل همان مساله زمانبندی جریان کارگاهی کلاسیک محسوب می‌شود و در مرحله دوم هم زمانبندی مونتاژ انجام می‌شود. زمان‌بندی برای تولید قطعات و برنامه‌ریزی برای عملیات مونتاژ به‌طور مستقل ممکن است نتیجه ایده‌آلی برای کل سیستم نداشته باشد (یوکویاما و سانتوز ۲۰۰۵؛ فتاحی و همکاران، ۲۰۱۳). بنابراین این دو مساله زمانبندی باید به طور همزمان مورد توجه قرار بگیرند (یوکویاما، ۲۰۰۸).

اولین مطالعه در مورد مسئله زمان‌بندی جریان کارگاهی مونتاژ دو مرحله‌ای توسط لی و همکاران (۱۹۹۳) انجام شد. آنها TSAFP سه ماشینه را برای به حداقل رساندن زمان تکمیل آخرین کار[2] مورد مطالعه قرار دادند و همچنین ثابت کردند که مساله با دو ماشین در مرحله اول و یک ماشین در مرحله دوم NP-hard است. آنها یک روش تقریبی و یک الگوریتم مبتنی بر شاخه و کران برای این مساله پیشنهاد دادند. پاتس و همکاران (۱۹۹۵) همان مساله مقاله لی و همکاران (۱۹۹۳) را برای حالت m ماشین تولیدی و یک دستگاه مونتاژ گسترش دادند و یک الگوریتم ابتکاری برای حل آن ارائه کردند. حریری و پاتس (۱۹۹۷) برای همین مساله m ماشین تولیدی یک روش حل شاخه و کران پیشنهاد دادند. چنگ و وانگ (۱۹۹۹) مساله جریان کارگاهی مونتاژ را با ساختار خاصی در نظر گرفتند تا بتوانند زمان تکمیل

1 .Two-stage assembly flow shop problem
2 .Makespan

آخرین کار را به حداقل برسانند. در مدل آن‌ها اولین ماشین دو نوع قطعه تولید می‌کرد و ماشین دوم قطعات را به محصول نهایی مونتاژ می‌کرد. یوکویاما (۲۰۰۸) فعالیت‌های آماده-سازی و متوسط زمان تکمیل برای همه محصولات را در مساله TSAFP در نظر گرفتند. در این مساله تعدادی مشخص از یک محصول تولید می‌شود که هر محصول از چندین قطعه مختلف تشکیل شده است. در ابتدا، قطعات در یک سطح جریان کارگاهی با m ماشین مختلف تولید می‌شود و سپس آن‌ها در یک ایستگاه تک ماشینه به محصولات نهایی مختلف مونتاژ می‌شوند. آنها یک برنامه ریزی شبه دینامیکی و یک روش شاخه و کران برای حل این مساله پیشنهاد دادند. یوکویاما (۲۰۰۱) و یوکویاما و سانتوز (۲۰۰۵) مساله TSAFP را بررسی‌کردند و یک روش حل برای به‌دست آوردن راه حل اپسیلن-بهینه براساس یک الگوریتم شاخه و کران برای آن توسعه دادند.

در ادبیات تحقیق، مساله TSAFP با معیارهای بهینه سازی مختلف در نظر گرفته شده است. برای مثال لی و همکاران (۱۹۹۳)، پاتس و همکاران (۱۹۹۵)، کولاماس و کایپاریسیس (۲۰۰۱) و الله وردی و الانزی (۲۰۰۷) مساله TSAFP را با حداقل کردن زمان تکمیل آخرین کار مطالعه کردند. در مقاله‌های الله‌وردی و الانزی (۲۰۰۶) و الانزی و الله‌وردی (۲۰۰۷) نیز مساله با معیار مغایرت زمانی تحویل یا همان میزان دیرکرد[1] در نظر گرفته شد. توزکاپان و همکاران (۲۰۰۳) این مساله را برای کمینه‌سازی مجموع زمان تکمیل وزن دار در نظرگرفتند. ماژگیر و همکاران (۲۰۱۳) مساله را با مجموع زمان تکمیل آخرین کار و متوسط زمان تکمیل هر کار مطرح کردند. همچنین در تحقیقی جدید غلامی و همکاران (۱۳۹۷) مسئله TSAFP را با هدف کمینه‌سازی زمان تکمیل کارها مورد مطالعه قرار دادند. آن‌ها یک مدل ریاضی برنامه ریزی عدد صحیح مختلط را برای حل این مسئله ارائه دادند. همچنین برای بهینه سازی این مسئله در ابعاد بزرگ از دو الگوریتم ژنتیک و رقابت استعماری استفاده کردند.

در این مقاله مساله زمان‌بندی جریان کارگاهی مونتاژ دومرحله‌ای با در نظر گرفتن زمان‌های آماده‌سازی مستقل از توالی مورد مطالعه قرار گرفته است. تابع هدف مسئله تحقیق

1 .Lateness

نیز کمینه‌سازی مجموع زمان تکمیل کل کارها یا سفارشات است. با توجه به NP-hard بودن مسئله مورد مطالعه و همچنین قرارگیری آن در طبقه مسائل بهینه‌سازی ترکیباتی، توسعه الگوریتم‌های فراابتکاری برای بهینه‌سازی این مسئله به خصوص در ابعاد بزرگ گزینه بسیار مناسبی می‌باشد. تاکنون بسیاری از روش‌های فراابتکاری برای حل مسائل بهینه‌سازی ترکیباتی پیشنهاد شده‌اند. برای مثال می‌توان به کاربرد این روش‌ها نظیر الگوریتم ژنتیک[۱](GA) (غلامی و همکاران، ۱۳۹۷)، شبیه‌سازی تبرید[۲] (SA) (ایگلز، ۱۹۹۰)، تکامل تفاضلی[۳] (DE) (استورن و پرایس، ۱۹۹۷)، جستجوی همسایگی متغیر[۴](VNS) (ملادنویچ و هنسن، ۱۹۹۷)، جستجوی ممنوعه[۵] (TS) (گلاور، ۱۹۸۶)، بهینه‌سازی کلونی مورچگان[۶] (ACO) (دوریگو و گامباردلا، ۱۹۹۷)، الگوریتم ایمنی مصنوعی (AIA) [۷] (باقری و همکاران، ۲۰۱۰) و الگوریتم رقابت استعماری[۸] (ICA) (آتشپز و لوکاس، ۲۰۰۷) در حل مسائل بهینه سازی اشاره کرد. در در این مقاله سه الگوریتم فراابتکاری جستجوی همسایگی متغیر موازی[۹] (PVNS)، ایمنی‌مصنوعی و شبیه‌سازی تبرید برای حل مسئله تحقیق پیشنهاد می‌شود. همچنین نتایج الگوریتم‌های پیشنهادی با یکدیگر مقایسه می‌شوند.

ادامه این مقاله به شرح زیر است. در بخش ۲، تعریف مساله همراه با یک مثال عددی ارائه شده است. الگوریتم‌های فراابتکاری پیشنهادی در بخش۳ معرفی‌شده‌اند. در بخش ۴، پارامترهای الگوریتم‌ها تنظیم می‌شوند. همچنین در این بخش نتایج الگوریتم‌ها و مقایسات آماری عملکرد آن‌ها ارائه می‌شود. بخش ۵ مقاله با اشاره به نتایج تحقیق و ارائه پیشنهادات برای تحقیقات آتی، مقاله را به پایان می‌رساند.

1 .Genetic algorithm
2 .Simulated annealing
3 .Differential evolution
4 .Variable neighborhood search
5 .Tabu search
6 .Ant colony optimization
7 .Artificial immune algorithm
8. Imperialist competitive algorithm
9 .Parallel variable neighborhood search

تعریف مساله

در این مقاله، زمان‌بندی یک مساله جریان کارگاهی مونتاژ دومرحله‌ای(TSAFP) با در نظر داشتن زمان‌های آماده‌سازی مستقل از توالی مطالعه می‌شود. در این مسئله یک سفارش n تایی از یک نوع محصول با قطعات مختلف (g قطعه) وجود دارد که هر قطعه، m عملیات مختلف دارد. در مرحله اول عملیات‌های مربوط به هر قطعه در یک سطح جریان کارگاهی با m ماشین مختلف انجام می‌شود. هر قطعه jدر ماشین i ام دارای زمان پردازش$P_{j,i}$ است. در این مرحله برای تولید هر قطعه باید زمان آماده سازی ماشین را در نظر گرفت. درصورتی که دو قطعه مختلف از یک سفارش و یا از سفارش‌های دیگر در توالی تولیدی پشت سر هم باشند، باید مراحل آماده سازی ماشین صورت گیرد. زمان‌های آماده سازی ماشین‌ها مستقل از توالی قطعات است و برابر با $S_{j,i}$ (زمان آماده سازی قطعه jروی ماشین i) در نظر گرفته شده است. این زمان آماده سازی با توجه به شماره قطعات و ماشین می‌تواند متفاوت باشد، بدین صورت که اگر ماشین برای انجام عملیات یک قطعه خاص تنظیم گردد در ادامه ماشین باید مجددا برای پردازش قطعه بعدی تنظیم شود. تنها زمانی به آماده سازی ماشین برای پردازش احتیاج نیست که یک قطعه یکسان برای دو سفارش مختلف پشت سر هم روی یک ماشین باشند. بعد از تولید کامل این قطعات، در مرحله‌ی دوم عملیات مونتاژ قطعات روی تک ماشین انجام می‌شود. با مونتاژ این قطعات، مرحله تولید یک محصول (یکی از سفارش‌ها) را می‌توان تکمیل کرد. زمانی می‌توان مونتاژ یک سفارش را شروع کرد که تمامی قطعات آن در مرحله نخست تولید شده باشد. قطعات هر سفارش که زودتر در مرحله اول تکمیل شود، مونتاژ آن سفارش نیز در مرحله دوم زودتر انجام می‌شود. تابع هدف مسئله نیز کمینه‌سازی مجموع زمان تکمیل کل کارها یا سفارشات است. در این مقاله مفروضات زیر برای مسئله TSAFP نظر گرفته شده‌اند:

(۱) قطعات و ماشین‌ها در زمان صفر در دسترس هستند.

(۲) قطع پردازش یک کار یا یک قطعه چه در مرحله اول (تولید) و چه در مرحله دوم (مونتاژ) مجاز نیست.

(۳) هر ماشین به طور همزمان فقط می‌تواند یک قطعه را پردازش کند.

(۴) هر قطعه به طور همزمان می‌تواند روی یک ماشین پردازش شود.

(۵) محصولات مشابه هستند.

(۶) زمان پردازش هر قطعه مشخص است.

(۷) ماشین‌ها دائما در دسترس هستند.

(۸) زمان حمل و نقل بین ماشین‌ها در نظر گرفته نمی‌شود.

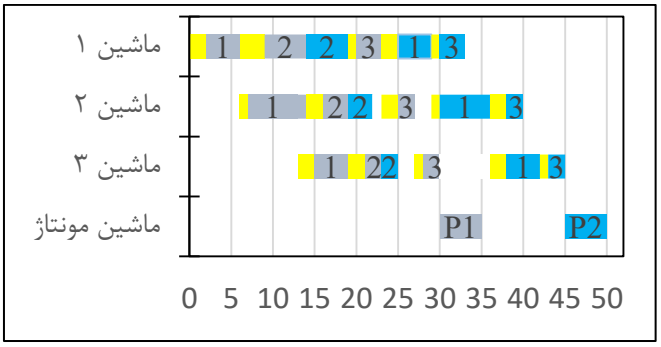
حال برای روشن تر کردن مساله مورد نظر، یک مثال عددی ارائه می‌شود. مساله‌ای با دو سفارش از یک محصول را در نظر بگیرید. محصول متشکل از سه قطعه است. بنابراین، در مجموع ۶ قطعه در مرحله‌ی اول تولید شود، دو قطعه شماره ۱، دو قطعه شماره ۲، دو قطعه شماره ۳. همچنین هر قطعه نیازمند سه عملیات برای تکمیل است. در نتیجه، در مرحله اول سه ماشین وجود دارد. بعد از تکمیل شدن قطعات در این مرحله، آنها در مرحله بعد توسط یک ماشین مونتاژ به محصولات نهایی مونتاژ می‌شوند. زمان‌های پردازش و زمان‌های آماده‌سازی در جدول ۱ ارائه شده است. همچنین زمان هر عملیات مونتاژ ۵ P_A= است. نحوه نمایش یکی از جواب‌های موجه برای این مثال عددی در شکل ۱ نمایش داده شده است. اعداد در این شیوه نمایش جواب نشان دهنده شماره قطعات و تکرار آن‌ها بیانگر شماره قطعات مشابه از سفارشات مختلف یک نوع محصول می‌باشد. ترتیب قرار گیری اعداد نشان دهنده ترتیب انجام قطعات مربوط به سفارشات است که روی تمام ماشین‌ها یکسان است. شکل ۲ نمودار زمانی گانت را برای این جواب نشان می‌دهد.

۱	۲	۲	۳	۱	۳

شکل ۱: نحوه نمایش جواب یک مسئله با ۲ سفارش و ۳ قطعه

جدول (۱): زمان های پردازش و آماده سازی

قطعات	زمان پردازش قطعات روی ماشین‌ها			زمان آماده سازی قطعات روی ماشین‌ها		
	M1	M2	M3	M1	M2	M3
۱	۴	۶	۴	۲	۱	۲
۲	۵	۳	۲	۳	۲	۲
۳	۳	۲	۲	۱	۲	۱



شکل ۲: نمودار گانت یک جواب موجه برای مسئله جدول ۱

با توجه به توضیحات، در شکل ۱ می‌بایست شماره هر قطعه در بردار نمایش جواب دوبار تکرار شود. زیرا دو سفارش یکسان از یک محصول را در این مثال داریم. عدد ۱ موجود در اولین خانه بیانگر قطعه اول از سفارش اول است و عدد ۱ موجود در خانه پنجم بیانگر قطعه اول از سفارش دوم است. در شکل ۲ بخش‌های خاکستری رنگ نشان‌دهنده مراحل پردازش قطعات و مونتاژ سفارش شماره ۱ و بخش‌های آبی رنگ نشان‌دهنده مراحل پردازش قطعات و مونتاژ سفارش شماره ۲ می‌باشند. همچنین بخش‌های زرد رنگ زمان‌های آماده‌سازی ماشین‌ها را نشان می‌دهد. همانطور که در شکل ۲ مشخص است، هر قطعه از هر سفارش نیاز به انجام ۳ عملیات دارد تا تولید شود. پس از آنکه ۳ قطعه مربوط به هر سفارش آماده گردید، بلافاصله عملیات مونتاژ نهایی آن سفارش بر روی ماشین مونتاژ آغاز می‌گردد.

الگوریتم‌های فراابتکاری پیشنهادی

مسئله TSAFP از مسائل بهینه‌سازی ترکیباتی پیچیده است و از آن جایی که این مسئله در دسته مسائل NP-hard قرار می‌گیرد (پاتز و همکاران، ۱۹۹۵)، بهینه‌سازی و دستیابی به یک جواب با کیفیت مناسب برای آن بخصوص در مقیاس بزرگ مسئله با استفاده از روش‌های حل دقیق در یک زمان محاسباتی منطقی عملا غیر ممکن است. بنابراین در این مقاله سه الگوریتم فراابتکاری جستجوی همسایگی متغیر موازی، ایمنی مصنوعی و شبیه‌سازی تبرید برای حل این مسئله توسعه داده شده است. در ادامه به تشریح این الگوریتم‌ها می‌پردازیم.

الگوریتم جستجوی همسایگی متغیر موازی

ملادنویچ و هنسن (۱۹۹۷)، نخستین پژوهشگرانی بودند که الگوریتم جستجوی همسایگی متغیر (VNS) را توسعه دادند. این روش یکی از الگوریتم‌های فراابتکاری جدید بر پایه تغییرات سیستماتیک ساختارهای همسایگی می‌باشد که برای جستجوی جواب‌های بهینه در فضای مسائل بهینه سازی ترکیباتی کاربرد دارد. این تغییرات از طریق تغییر موقعیت از یک همسایگی به همسایگی دیگر جواب در هنگام جستجو در فضای جواب صورت می‌گیرد. در واقع الگوریتم VNS در قالب یک فرایند ترتیبی شامل فرآیندهای ارتعاش و جستجوی محلی، ساختارهای همسایگی متنوع را به کار می‌گیرد و فرآیند جستجوی جواب را در نواحی مختلف از فضای جواب به شکل مناسبی انجام می‌دهد. اصول این الگوریتم براساس این موضوع می‌باشد که تغییرات ساختارهای همسایگی در هنگام جستجوی فضای جواب، از خطر افتادن در بهینه محلی در فرایند بهینه سازی مسائل ترکیباتی جلوگیری می‌کند. شبه‌کد پایه‌ای الگوریتم VNS در شکل ۳ نمایش داده شده است.

در به‌کارگیری الگوریتم‌های فراابتکاری برای حل مسائل بهینه‌سازی ترکیباتی، معمولا الگوریتم مدت زمان طولانی را برای دستیابی به یک جواب مناسب صرف خواهد کرد. این مشکل می‌تواند به علت عدم اجرای یک جستجوی جامع در فضای جواب باشد که در نتیجه موجب یک نقصان در دستیابی به جواب‌های بهینه یا نزدیک به بهینه و طولانی بودن زمان

جستجو می گردد. استفاده از فرآیند موازی سازی الگوریتم یک راه حل مناسب برای مواجهه با این مشکل می‌باشد (سوکلی و آیدین، ۲۰۰۷). موازی سازی الگوریتم‌های فراابتکاری به ما این امکان را می‌دهد تا جستجوهای چندگانه همزمان را با استفاده از تعدادی پردازشگر در طول هر تکرار از الگوریتم بر روی فضای جواب با هدف دسترسی به یک جستجوی کامل و سریع به کار گیریم. با توجه به اینکه در هر تکرار از فرآیند ترتیبی الگوریتمVNS ، وقت گیرترین بخش مربوط به فرآیند جستجوی محلی می باشد، این روش به وسیله موازی‌سازی این فرآیند در الگوریتمVNS زمان محاسباتی الگوریتم را کاهش می‌دهد که هر کدام از این جستجوها توسط یک پردازشگر انجام می‌شود. شبه‌کد الگوریتم جستجوی همایگی متغیر موازی (PVNS) ارائه شده در این مقاله برای مسئله TSAFP در شکل ۴ نمایش داده شده است.

الگوریتم PVNS ارائه شده از دو بخش اصلی حلقه داخلی و حلقه خارجی تشکیل شده است. در حلقه داخلی جستجوی فضای جواب صورت می‌گیرد در حالی که حلقه خارجی کنترل شرط توقف الگوریتم را برعهده دارد. در هر تکرار از حلقه داخلی الگوریتم، هر پردازشگر با توجه به جواب ورودی یک تکرار از فرایند ترتیبی VNS شامل فرآیندهای ارتعاش و جستجوی محلی را اجرا می‌کند. نماد k نشان‌دهنده شمارنده تکرار حلقه داخلی و نوع ساختار همسایگی به کار گرفته شده در فرآیند ارتعاش (N_k^s) و نماد n شمارنده تکرار حلقه جستجوی محلی و نماد l نوع ساختار همسایگی به کار گرفته شده در فرآیند ایجاد جواب تصادفی جدید (N_l^s) در مرحله جستجوی محلی می‌باشد.

1. Initialization

Define the set of neighborhood structures N_k, for $k = 1,...,k_{max}$ that will be used in the search;

Choose a stopping condition; and find the initial solution x;

2. Repeat until stopping condition is met:

Set $k \longleftarrow 1$;

3. Repeat the following steps until $k > k_{max}$:

3.a. Shaking: Generate point x' at random from the k^{th} neighborhood of x ($x' \in N_k(x)$);

3.b. Local search: Apply some local search methods with x' as initial solution

to obtain a local optimum given by x'';

3.c. Neighborhood change:

if the local optimum x'' is better than x then $x \longleftarrow x''$ and continue the search

with N_1 ($k \longleftarrow 1$); Otherwise, set $k \longleftarrow k+1$;

شکل ۳: شبه کد الگوریتم VNS

Initialization:
Select the set of neighborhood structures N_k^s, for k=1,...,$k_{\max}$ *and* N_l^s, for l = 1,...,$l_{\max}$ that will be used in the shaking and local search procedure, respectively; find Initial solution x; set number of processors (n_{pr}); determine number of neighborhood seach in the local search procedure; choose the stopping condition.

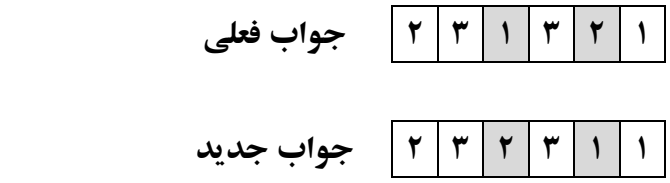
Repeat (external loop)
set $k \leftarrow 1$;
Repeat (internal loop)
for each processor pr(h) h=1,...,n_{pr} do in parallel
Shaking procedure
Generate random soluion $x'_{(h)}$ by N_k^s from k^{th} neighborhood of x;
Local search procedure
Get solution $x'_{(h)}$;
set $n \leftarrow 1$ and $l \leftarrow 1$;
for i=1:$n_{\max}$
Generate random soluion xp by N_l^s from l^{th} neighborhood of x;
if f(xp) ≤ f($x'_{(h)}$) then $x'_{(h)} \leftarrow xp$ and $l \leftarrow l$;
else select the random integer number R in the range $(1, l_{\max})$ and $l \leftarrow R$;
endif
end for
$\text{spr}_{(h)} \leftarrow x'_{(h)}$; (obtained solution by h^{th}processor in the local search procedure)
endfor
Updating
The best solution is selected among the obtaines solution of the processors and it place in x'';
if $f(x'') < f(x)$ then $x \leftarrow x''$ and $k \leftarrow 1$; else $k \leftarrow k+1$;
endif
until $k > k_{\max}$
until the stopping condition is reached

شکل٤: شبه کد الگوریتم PVNS توسعه داده شده

همچنین در این مقاله دو ساختار جستجوی همسایگی "معاوضه" و "افزودن" برای بهبود جواب‌ها در مرحله ارتعاش و جستجوی محلی استفاده می‌شود.

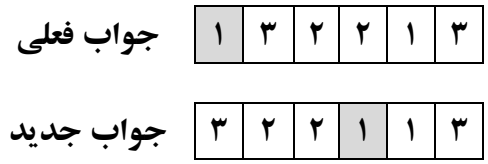
۱) ساختار همسایگی معاوضه: در این عملگر، موقعیت‌های دو قطعه در جواب به تصادف انتخاب و تعویض می‌شوند. شکل ۵ نحوه عملکرد عملگر معاوضه را بر روی یک جواب

موجه برای یک مسئله با ۲ سفارش از یک محصول ۳ قطعه ای نشان می‌دهد. فرض کنید موقعیت‌های انتخابی در جواب ۳ و ۵ هستند. سپس شماره‌های قطعات موجود در این دو موقعیت عوض می‌شوند و یک جواب جدید ایجاد می‌شود.



شکل ۵: نحوه عملکرد ساختار همسایگی معاوضه

۲) ساختار همسایگی افزودن: در این عملگر، یک قطعه به تصادف در جواب انتخاب می‌شود و در یک خانه جدید دیگر به شکل تصادفی قرار می‌گیرد. شکل ۶ نحوه عملکرد عملگر افزودن را نشان می‌دهد. فرض کنید که موقعیت انتخاب شده تصادفی موقعیت ۱ باشد و موقعیت انتخاب شده تصادفی برای ورود مجدد موقعیت ۴ است. شماره قطعه ۱ به موقعیت ۴ جابجا می‌شود و سپس بقیه قطعات با توجه به ترتیب قبلی و موقعیت جا به جایی انجام شده، در خانه‌های دیگر جواب قرار می‌گیرند.



شکل ٦: نحوه عملکرد ساختار همسایگی افزودن

در حلقه داخلی الگوریتم مورد نظر، در هر بار اجرای فرآیند ارتعاش و در هر بار ایجاد جواب جدید در مرحله جستجوی محلی از یکی از دو ساختار همسایگی معاوضه و یا افزودن که به شکل تصادفی از میان این دو عملگر انتخاب می‌شود، استفاده می‌شود. همچنین با توجه به شمارنده‌های k و l این دو عملگر با تعداد تکرارهای بیشتر در ایجاد یک جواب تصادفی جدید به کار می‌روند. برای مثال اگر در فرآیند ارتعاش عملگر معاوضه برای ایجاد تغییرات روی جواب انتخاب شده باشد و در تکرار دوم از حلقه داخلی باشیم (N_2^s, k=2)، این عملگر برای ایجاد یک جواب جدید ۲ بار روی جواب اعمال می‌شود و جواب خروجی جوابی است که هر ۲ تغییر روی آن صورت گرفته باشد. همچنین برای مثال اگر در فرآیند جستجوی محلی و در یکی از تکرارها عملگر افزودن برای ایجاد تغییرات روی جواب انتخاب شده باشد و ساختار همسایگی چهارم برای اعمال تغییرات روی جواب به شکل تصادفی انتخاب شده باشد (N_4^s, l=4)، این عملگر برای ایجاد یک جواب جدید ۴ بار روی جواب اجرا می‌شود و جواب خروجی جوابی است که هر ۴ تغییر روی آن صورت گرفته باشد. در این مقاله $k_{\max}$ برابر با ۳ و $l_{\max}$ برابر با ۵ در نظر گرفته شده است. در پایان هر تکرار از حلقه داخلی از میان جواب‌های نهایی به دست آمده برای پردازشگرها، بهترین جواب انتخاب می‌شود و در x'' قرار می‌گیرد. سپس x'' با بهترین جواب به دست آمده تا آن مرحله یعنی x مقایسه می‌شود و در صورتی که $f(x'') < f(x)$ آن گاه $x \leftarrow x''$ و $k \leftarrow 1$. در غیر این صورت مقدار x تغییر نمی کند و تکرار بعدی از حلقه داخلی انجام می‌گردد. لازم به ذکر است در شروع هر از تکرار حلقه داخلی الگوریتم، جواب ورودی تمام پردازشگرها برابر با x قرار می‌گیرد. شرط توقف الگوریتم PVNS ماکزیمم زمان محسباتی تعیین شده است که با توجه به ابعاد مسئله می‌تواند تغییر کند.

الگوریتم ایمنی مصنوعی

الگوریتم ایمنی مصنوعی (AIA) که نمونه ای از روش‌های بهینه‌سازی هوشمند است، با بهره گیری از ایده سیستم‌های ایمنی طبیعی توسعه داده شده است. در این راستا طی سالیان اخیر محققان بسیاری در ارتباط با سیستم‌های ایمنی مصنوعی، تحقیقات ارزشمندی را انجام داده

اند (زندیه و همکاران، ۲۰۰۶؛ داسگوپتا، ۲۰۰۲؛ کاسترو و تیمیس، ۲۰۰۲). سیستم ایمنی مصنوعی، به عنوان یک سیستم محاسباتی که از توابع، اصول و مدل‌های ایمنی الهام گرفته-است، در حل مسائل بهینه سازی ترکیباتی بکار می‌رود (باقری و همکاران، ۲۰۱۰؛ شانگ و همکاران، ۲۰۱۹). شبه‌کد پایه‌ای الگوریتم ایمنی مصنوعی به صورت زیر است:

- n جواب موجه (آنتی بادی) به شکل تصادفی تولید می‌شود.
- میزان سازگاری جواب‌ها ارزیابی می‌گردد.
- جواب‌هایی با سازگاری بالا جهت تکثیر انتخاب می‌شوند (نرخ تکثیر هر جواب با سازگاری آن نسبت مستقیم دارد).
- بر روی جواب‌های تولید شده فرآیند جهش صورت می‌گیرد (میزان جهش بر روی هر جواب با سازگاری آن نسبت معکوس دارد).
- n جواب با توجه به استراتژی انتخاب الگوریتم تعیین می‌شوند و به نسل بعدی می‌روند. تا زمانی که شرط توقف تعیین شده رخ ندهد، روند الگوریتم تکرار می‌شود و ادامه می‌یابد.

شبه کد الگوریتم AIA توسعه داده شده در این مقاله برای مسئله TSAFP در شکل ۷ نمایش داده شده است. الگوریتم ارائه شده شامل دو بخش اصلی انتخاب کلونی[1] و بلوغ سازگاری[2] می‌باشد. مطابق شکل ۷ در بخش انتخاب کلونی ابتدا جواب‌های (آنتی‌بادی‌های) با سازگاری بالاتر از میان جمعیت جواب‌ها (سایز جمعیت جواب‌ها یا آنتی بادی‌ها برابر با n_{ab} است) انتخاب می‌شوند. تعداد جواب‌های انتخاب شده در این مرحله معادل با نصف اندازه جمعیت جواب‌ها ($n=\frac{n_{ab}}{2}$) می‌باشد. پس از مشخص شدن n جواب با بالاترین

1. Clonel selection
2. Affinity muturation

سازگاری، با بهره‌گیری از روش انتخاب رقابتی دودویی تعداد $n_{ab}-1$ جواب تکثیر می‌شوند و به استخر جفت گیری منتقل می‌شوند. در مرحله پایانی این بخش بهترین جواب موجود در میان جمعیت از نقطه نظر سازگاری به استخر جفت‌گیری منتقل می‌شود تا جمعیت جواب‌های انتخابی برابر با n_{ab} به دست آید. لازم به ذکر است احتمال وارد شدن و کلون شدن هر جواب رابطه مستقیم با میزان سازگاری آن دارد. میزان سازگاری هر جواب (آنتی بادی) از طریق رابطه زیر محاسبه می‌گردد (لو و چوئه، ۲۰۰۹):

$$\text{Affinity}(\text{ab}_i) = \frac{\min(obj(ab)_i \mid i = 1,...,n_{ab})}{obj(ab_i)} \qquad (۱)$$

جایی که $obj(ab)_i$ نشان دهنده تابع هدف iامین جواب (آنتی بادی) و n_{ab} نشان دهنده تعداد جمعیت جواب‌ها (آنتی بادی‌ها) می‌باشد. همچنین مراحل بخش بلوغ سازگاری شامل فرآیندهای فرا جهش و اصلاح گیرنده می‌باشد. در بخش بلوغ سازگاری، بر روی تمام کلونی جواب‌ها فرآیند جهش انجام می‌گیرد. تغییرات انجام شده روی جواب‌ها بستگی به مقدار سازگاری هر جواب دارد. بر روی جواب‌هایی با سازگاری بالاتر تغییرات با نرخ کمتر و بر روی جواب‌هایی با سازگاری پایین‌تر تغییرات با نرخ بیشتر صورت می‌گیرد. ما از معیار زیر برای کنترل فرآیند فراجهش استفاده می‌کنیم. در این معیار جواب‌ها (آنتی بادی‌ها) تحت فرآیند فراجهش با نرخ پایین قرار می‌گیرند (باقری و همکاران، ۲۰۱۰):

$$\text{if } \frac{obj(ab)_i - obj(best \text{ antibody})}{obj(best \text{ antibody})} \le 0.1 \qquad (۲)$$

در غیر این صورت بر روی جواب (آنتی بادی) فرآیند جهش با نرخ بالاتر اجرا می‌گردد.

Initialization:
Set the size of antibody population (n_{ab}); generate initail antibody population, Deterrmine the mutation operators; set the number of exchangable antibodiesreceptor editing (n_{re}) ; set *pop* ← initail antibody population.

Repeat

Clonel selection and expansion

Calculate objective function (makespan) and affinity values of each antibody of *pop*;

Slelect $\left(n=\frac{n_{ab}}{2}\right)$ antibodies from the *pop* with the highest affinity;

Prolifrate (n_{ab}-1) clones (copies) from the selected antibodies by using binary tournament rule (choose two antibodies from m antibodies randomly and select the antibody with higher affinity) and transfer clones to mutating pool (The higher the affinity, the greater the number of copies in mutating pool, and vice versa);

Transfer the best antibody to mutating pool; (mutating pool size=n_{ab})

Affinity muturation

Somatic hypermutation: Mutate clones in the mutating pool with a rate proportional to their affinity.The higher the affinity, the less the hypermutation rate, and vice versa;

Add the mutated clones (new antibodies) to the current *pop* and create *pop′*; (*pop′*size=$2\times n_{ab}$)

Receptor editing: Replace n_{re} antibodies with the lowest affinity with new ones in the *pop′*;

Updating next population

Copy the best antiody of *pop′* to the next *pop*;

Select n_{ab}-1 antibodies from *pop′* by using a suitable selection strategy and copy them into the next *pop*;

until the stopping condition is reached

شکل ۷: شبه کد الگوریتم AIA توسعه داده شده

ساختارهای همسایگی به کار گرفته شده در بخش قبلی به عنوان عملگر جهش برای این الگوریتم مورد استفاده قرار گرفته است. پس از اتمام فرآیند فراجهش، کلونی جواب‌های جهش داده شده به جمعیت فعلی اضافه می‌شوند، بنابراین جمعیت *pop′* با تعداد ($pop'\text{size}=۲\times n_{ab}$) ایجاد می‌شود. در ادامه جواب‌هایی با سازگاری پایین تر از این جمعیت حذف و جواب‌های جدید جایگزین آن می‌شوند. در این مکانیسم از یک الگوی طبیعی موجود در سیستم ایمنی بدن بهره گرفته شده است که اصلاح گیرنده [1] نام دارد. جستجوی فضاهای جدید توسط این عملگر به الگوریتم در فرار از نقطه بهینه محلی کمک می‌کند. شرط توقف الگوریتم AIA ماکزیمم زمان محاسباتی تعیین شده است که با توجه به ابعاد مسئله می‌تواند تغییر کند.

1 . Receptor editing

شبیه‌سازی تبرید

شبیه‌سازی تبرید (SA) یک الگوریتم فراابتکاری مبتنی بر جستجوی محلی است که از ساز و کار تبرید مواد جامد الهام گرفته است و این توانمندی را دارد که نقطه بهینه محلی را با پذیرش مشروط راه‌حل‌های بدتر ترک کند. این روش جستجو از یک جواب اولیه تصادفی sشروع می‌کند. سپس از این جواب s یک جواب جدید s' با استفاده از عملگر ساختار همسایگی ایجاد می‌کنیم. ساختارهای همسایگی به‌کارگرفته شده در الگوریتمPVNS برای الگوریتم SA نیز مورد استفاده قرار گرفته است. در ادامه تابع هدف این جواب را ارزیابی می‌کنیم. اگر داشته باشیم $\Delta= f(s') - f(s) \leq ۰$ آن گاه s' به عنوان جواب جدید پذیرفته می‌شود. در غیر این صورت با احتمال ارائه شده توسط $exp(-\Delta/T)$ جواب جدید پذیرفته شده می‌شود. پارامتر T یک پارامتر کنترلی به عنوان درجه حرارت است. بعد از اتمام فرایند جستجو در هر تکرار از الگوریتم، با استفاده از برنامه خنک سازی دما کاهش می‌یابد و این فرایند تا زمان توقف الگوریتم تکرار می‌شود. در این مقاله برای کاهش دما در الگوریتم SA از یک تابع کاهش هندسی به صورت رابطه ذیل استفاده شده است (یزدانی و همکاران، ۱۳۹۳):

$$T_{i+1} = \alpha \times T_i \qquad (۳)$$

جایی که $0 < \alpha < 1$ پارامتر کاهش دهنده دما می‌باشد و مقدار این ضریب برابر با ۰/۹ تنظیم شده است. شکل ۸ شبه‌کد پایه‌ای الگوریتم SA را نشان می‌دهد که در این مقاله استفاده شده است. شرط توقف الگوریتم SA ماکزیمم زمان محاسباتی تعیین شده است که با توجه به ابعاد مسئله می‌تواند تغییر کند.

```
Procedure :SA Algorithm
Initialization mechanism (initial solution (
While the stopping criterion is not met do
    Move mechanism
    Acceptance mechanism
    Temperature reduction
Endwhile
```

شکل ۸: شبه کد الگوریتم SA

مطالعات محاسباتی

این بخش عملکرد الگوریتم‌های ارائه شده را ارزیابی می‌کند. دو مجموعه مثال نمونه در سایزهای کوچک و بزرگ تولید می‌شود. در مثال‌های کوچک و بزرگ، نتایج الگوریتم‌ها با یکدیگر مقایسه می‌شوند. برای تولید مثال‌های کوچک، یک مجموعه از ۱۲ مساله از ترکیبات n (تعداد کارها یا سفارشات)، g (تعداد قطعات) و m (تعداد عملیات‌های مورد نیاز برای قطعات روی ماشین‌ها) درنظر گرفته می‌شود:

m= ۳و۵ g= ۲و۴ n=۲و۴و۶

به عبارت دیگر یک مثال برای هر ترکیب n، m و g تولید می‌شود. برای مثال‌های بزرگ، یک مجموعـه از ۱۲ مسالـه با ترکیبات زیر در نظـر گرفتـه می‌شود:

m= ۵و۷ g= ۴و۶ n=۱۰و۳۰و۵۰

زمان‌های پردازش و زمان‌های آماده سازی به صورت تصادفی از یک توزیع یکنواخت بین (۱,۹۹) تولید می‌شوند. زمان عملیات مونتاژ نهایی برای همه سفارشات در تمامی مسائل از یک توزیع یکنواخت بین (۱,۱۰) تولید می‌شوند. الگوریتم‌ها نیز در نرم افزار متلب Matlab R2016b پیاده‌سازی شده است و در یک کامپیوتر core i5 و ۸ GB حافظه اجرا می‌شوند. در این مقاله برای مقایسه الگوریتم‌ها فراابتکاری ارائه شده، از معیار درصد انحراف نسبی[1] (RPD) استفاده شده است. RPD با استفاده از رابطه زیر بدست می‌آید (یزدانی و همکاران، ۲۰۱۳):

$$RPD = \frac{Alg_{\text{sol}} - Min_{\text{sol}}}{Min_{\text{sol}}} \times ۱۰۰ \qquad (۴)$$

که در آن Min_{sol} بهترین جواب به‌دست آمده از میان الگوریتم‌ها برای مساله است و بهترین جواب به‌دست آمده از هر الگوریتم Alg_{sol} نام دارد. لازم به ذکر است به دلیل ماهیت تصادفی الگوریتم‌های فراابتکاری، هر الگوریتم را برای هر مسئله ۵ بار اجرا می‌کنیم. پس از

1.Relative percentage deviation

اتمام اجراهای هر سه الگوریتم بر روی مسئله مورد مطالعه و مشخص شدن بهترین جواب به دست آمده (Min_{sol})، مقدار RPDرا برای نتیجه ۵ تکرار اجرای هر الگوریتم بر روی مسئله مورد نظر به شکل جداگانه دست می‌آوریم و سپس میانگین RPDها را ($\overline{\text{RPD}}$) محاسبه می‌کنیم که این معیار به عنوان نتیجه نهایی الگوریتم بر روی مسئله مورد نظر در نظر گرفته می‌شود. در ادامه این محاسبات برای تمامی مسائل انجام می‌شود و $\overline{\text{RPD}}$ به عنوان مبنای مقایسات الگوریتم‌ها تعیین می‌شود.

تنظیم پارامترها

همانطور که مشخص است کارایی الگوریتم‌های فرا ابتکاری تا حد زیادی به تنظیم صحیح پارامترهای کنترلیشان مربوط است. لذا در این تحقیق، از طراحی آزمایش‌ها به روش تاگوچی، به منظور تنظیم پارامترهای الگوریتم‌های ارائه شده، استفاده می‌کنیم. برای الگوریتم PVNS پارامترهای n_{pr} (تعداد پردازشگرها) و $n_{\max}$ (تعداد جستجوهای همسایگی در مرحله جستجوی محلی)؛ برای الگوریتم AIA پارامترهای n_{ab} (اندازه جمعیت)، n_{re} (تعداد آنتی بادی‌های جایگزین شده در فرآیند اصلاح گیرنده) و استراتژی انتخاب استفاده شده جهت ایجاد نسل بعدی؛ و برای الگوریتم SA پارامترهای T_0)دمای اولیه(و $r_{\max}$ (تعداد جستجوهای همسایگی در هر دما) برای تنظیم پارامتر در نظر گرفته شده‌اند. موارد مربوط به سطوح مختلف تنظیم پارامتر در جداول ۲، ۳ و ۴ نمایش داده شده است. ۱۵ مثال مختلف در اندازه‌های مختلف تصادفی برای تنظیم پارامترها ساخته شده اند. نتایج محاسبات برای الگوریتم PVNS نشان می‌دهد مقدار n_{pr} برابر با ۱۰ و مقدار $n_{\max}$برابر با ۱۰۰ تعیین شده است. نتایج تنظیم پارامتر برای الگوریتم AIA بیانگر این است که مقدار n_{ab}برابر با ۷۵۰ و مقدار n_{re} برابر ۵۰ و استراتژی انتخاب استفاده شده انتخاب مسابقه‌ای تعیین شده است. همچنین طبق نتایج روش تاگوچی مقدار T_0 برای الگوریتم SA برابر با ۵۰ و مقدار $r_{\max}$ برابر با ۲۰۰ تعیین شده است.

جدول ۲: پارامترهای قابل تنظیم برای الگوریتم PVNS

فاکتورها	سطوح
تعداد پردازشگرها (n_{pr})	A(1)= ۵ A(2)= ۱۰ A(3)= ۱۵
تعداد جستجوهای همسایگی در مرحله جستجوی محلی ($n_{\max}$)	B(1)= ۱۰۰ B(2)= ۲۰۰ B(3)= ۳۰۰

جدول ۳: پارامترهای قابل تنظیم برای الگوریتم AIA

فاکتورها	سطوح
اندازه جمعیت (n_{ab})	A(1)= ۲۵۰ A(2)= ۵۰۰ A(3)= ۷۵۰ A(4)= ۱۰۰۰
تعداد آنتی بادی‌های جایگزین شده در فرآیند اصلاح گیرنده (n_{re})	B(1)= ۵۰ B(2)= ۷۵ B(3)= ۱۰۰ B(4)= ۱۲۵
استراتژی انتخاب	C(1) تصادفی C(2) انتخاب بهترین‌ها C(3) چرخه رولت C(4) انتخاب مسابقه ای

جدول ٤: پارامترهای قابل تنظیم برای الگوریتم SA

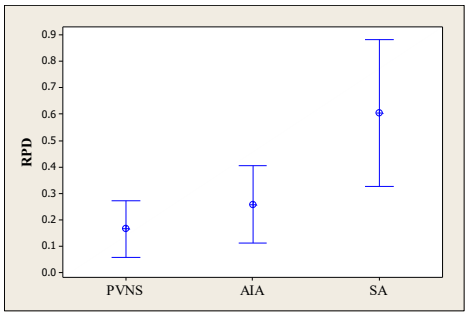
فاکتورها	سطوح
دمای اولیه (T_0)	A(1)= ۲۰ A(2)= ۵۰ A(3)= ۱۰۰
تعداد جستجوهای همسایگی در هر دما ($r_{\max}$)	B(1)= ۱۰۰ B(2)= ۲۰۰ B(3)= ۳۰۰

محاسبات عددی مسائل – اندازه کوچک

در ابتدا مثال‌های کوچک به وسیله الگوریتم‌های توسعه داده شده حل شده و نتایج در جدول ۵ نمایش داده شده است. با در نظر گرفتن معیار $\overline{\text{RPD}}$ میانگین نتایج الگوریتم PVNS(۰/۱۶۶۷) نسبت به الگوریتم‌های AIA (۰/۲۵۹۲) و SA (۰/۶۰۴۲) بهتر می‌باشد. همچنین نتایج آزمون تحلیل واریانس و روش حداقل اختلاف معنی‌دار با استفاده از نرم‌افزار Minitab نشان می‌دهد که از نقطه نظر معیار $\overline{\text{RPD}}$ در مسائل کوچک تفاوت معنی‌داری میان جواب‌های الگوریتم‌های PVNS و AIA وجود ندارد. اما هر دو الگوریتم عملکرد بهتری نسبت به SA دارند. شکل ۹ نمودار فاصله اطمینان ۹۵٪ نتایج الگوریتم‌ها را با توجه به معیار $\overline{\text{RPD}}$ برای مسائل کوچک نشان می‌دهد.

جدول (۵): متوسط $\overline{RPD}$ بدست آمده توسط الگوریتم‌ها از حل مسائل کوچک

ابعاد مسئله			الگوریتم ($\overline{RPD}$)		
n	g	m	PVNS	AIA	SA
۲	۲	۳	۰/۰	۰/۰	۰/۰
۲	۲	۵	۰/۰	۰/۰	۰/۰
۲	۴	۳	۰/۰	۰/۰	۰/۰
۲	۴	۵	۰/۰	۰/۰۹	۰/۳۱
۴	۲	۳	۰/۳۲	۰/۱۲	۰/۷۳
۴	۲	۵	۰/۲۱	۰/۲۸	۰/۶۴
۴	۴	۳	۰/۰	۰/۳۵	۰/۸۴
۴	۴	۵	۰/۳۶	۰/۵۹	۰/۶۷
۶	۲	۳	۰/۲۴	۰/۲۱	۰/۷۳
۶	۲	۵	۰/۰۹	۰/۳۲	۰/۹۶
۶	۴	۳	۰/۴۱	۰/۶۴	۱/۲۸
۶	۴	۵	۰/۳۷	۰/۵۱	۱/۰۹
Average $\overline{RPD}$			**۰/۱۶۶۷**	**۰/۲۵۹۲**	**۰/۶۰۴۲**



شکل۹: نمودار فاصله اطمینان ۹۵٪ نتایج الگوریتم‌ها با توجه به معیار $\overline{RPD}$ در حل مسائل کوچک

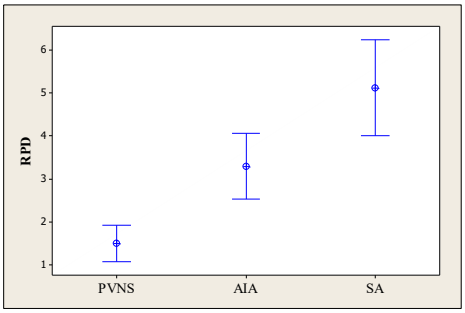
محاسبات عددی مسائل – اندازه بزرگ

در این بخش نتایج حل مثال‌های بزرگ توسط الگوریتم‌های پیشنهادی برای هر مسئله و الگوریتم گزارش شده است. جدول ۶ نتایج به دست آمده برای مسائل بزرگ را با توجه به

معیار $\overline{RPD}$ نشان می‌دهد. با در نظر گرفتن معیار $\overline{RPD}$ میانگین نتایج الگوریتم PVNS (۱/۵۰۶) نسبت به الگوریتم‌های AIA (۳/۲۹۴) و SA (۵/۱۱۸) بهتر می‌باشد. نتایج آزمون تحلیل واریانس و روش حداقل اختلاف معنی‌دار با بهره گیری از نرم‌افزار Minitab نشان دهنده برتری الگوریتم PVNS نسبت به دو الگوریتم AIA و SA در مسائل بزرگ است. همچنین مقایسات آماری نشان می‌دهد که الگوریتم AIA نسبت به الگوریتم SA عملکرد بهتری دارد. شکل ۱۰ نمودار فاصله اطمینان ۹۵٪ نتایج الگوریتم‌ها را با توجه به معیار $\overline{RPD}$ برای مسائل بزرگ نشان می‌دهد.

جدول ٦: متوسط $\overline{RPD}$ بدست آمده توسط الگوریتم‌ها از حل مسائل بزرگ

ابعاد مسئله			الگوریتم ($\overline{RPD}$)		
n	***g***	***m***	**PVNS**	**AIA**	**SA**
۱۰	٤	٥	۰/٥۲	۰/۹٥	۲/٤۳
۱۰	٤	۷	۰/٦۷	۱/۷٦	۲/۹۰
۱۰	٦	٥	۰/۷۲	۲/۲۷	۳/٦۳
۱۰	٦	۷	۱/۱۲	۲/۷۱	۳/۸٥
۳۰	٤	٥	۱/۸۷	۳/۳۲	٤/۳۱
۳۰	٤	۷	۱/۳۱	۲/۸۹	٤/٦٥
۳۰	٦	٥	۱/۹۷	۳/٦۸	٥/۷۲
۳۰	٦	۷	۲/۱۸	٤/۲۷	٦/۳٥
٥۰	٤	٥	۱/۱٤	۳/۹۳	٥/٥٤
٥۰	٤	۷	۲/۲۷	٤/۳٦	٦/۹۳
٥۰	٦	٥	۱/۸٥	٤/٤۹	۷/۸۱
٥۰	٦	۷	۲/٤٥	٤/٦۳	۷/۲۹
Average $\overline{RPD}$			**۱/٥۰٦**	**۳/۲۹٤**	**٥/۱۱۸**



شکل ۱۰: نمودار فاصله اطمینان ۹۵٪ نتایج الگوریتم‌ها با توجه به معیار $\overline{RPD}$ در حل مسائل بزرگ

نتیجه‌گیری و پژوهش‌های آتی

این مقاله، مسئله زمان‌بندی جریان کارگاهی مونتاژ دو مرحله ای (TSAFP) با وجود زمان‌های آماده‌سازی برای شروع تولید قطعات را با هدف به حداقل‌رساندن معیار بهینه سازی مجموع زمان تکمیل کل سفارشات مورد مطالعه قرار داد. دو الگوریتم فراابتکاری PVNS و AIA برای حل این مسئله مسئله توسعه داده شد. همچنین الگوریتم شناخته شده SA نیز برای حل آن تطبیق داده شد. برای ارزیابی عملکرد الگوریتم‌ها، یک مجموعه از مثال‌ها طراحی شد. در ادامه در اندازه‌های کوچک و بزرگ عملکرد الگوریتم‌ها مورد مقایسه قرار گرفت. در مسائل با اندازه کوچک مقایسات آماری نتایج نشان داد که الگوریتم‌های PVNS و AIA تفاوت معناداری از نقطه نظر معیار $\overline{RPD}$ ندارند ولی هر دو بهتر از الگوریتم SA عمل می‌کنند. اما با انجام مقایسات آماری از نقطه نظر معیار $\overline{RPD}$ در مسائل با اندازه بزرگ مشخص شد که الگوریتم PVNSبه طور معناداری از الگوریتم‌های AIA و SA عملکرد بهتری دارد. همچنین الگوریتم AIA در ابعاد بزرگ مسئله نسبت به الگوریتم SA عملکرد بهتری را نشان داد.

در این مقاله، مساله مورد مطالعه با مجموعه ای از مفروضات تعریف شد. برداشتن هر یک از این مفروضات و همچنین تعریف اهداف و محدودیت‌های جدید می‌تواند یک گام رو به

جلو برای ادبیات تحقیق این مسئله محسوب شود. به منظور تحقیقات آتی بر روی مسئله TSAFP پیشنهادات زیر ارائه می‌گردد:

- در نظر گرفتن زمان‌های جمع آوری و حمل قطعات تولید شده از مرحله اول به مرحله مونتاژ و بارگذاری آن‌ها بر روی ماشین مونتاژ
- در نظر گرفتن تابع هدف مجموع زمان دیرکرد با توجه به سفارش محور بودن مسئله
- در نظر گرفتن منابع دوگانه محدود نیروی انسانی و ماشین در زمان‌بندی مسئله مورد مطالعه

منابع

غلامی، حبیب رضا، مهدی زاده، اسماعیل، نادری، بهمن (۱۳۹۷) مدل‌سازی ریاضی و الگوریتم رقابت استعماری برای مسئله خط مونتاژ جریان کارگاهی، *چشم انداز مدیریت صنعتی*، شماره ۲۹، ۱۰۲-۸۴.

یزدانی، مهدی، زندیه، مصطفی، توکلی مقدم، رضا (۱۳۹۳) یک الگوریتم فراابتکاری ترکیبی برای مسئله زمان بندی کار کارگاهی منعطف با منابع دوگانه محدود انسان و ماشین، *مطالعات مدیریت صنعتی*، شماره ۳۳، ۴۳-۷۴.

Al-Anzi, F.S. &Allahverdi, A. (2007). A self-adaptive differential evolution heuristic for two-stage assembly scheduling problem to minimize maximum lateness with setup times, *European Journal of Operational Research,* 182(1), 80–94.

Allahverdi, A. & Al-Anzi, F.S. (2006). A PSO and a tabu search heuristics for assembly scheduling problem of the two stage distributed database application, *Computers and Operations Research*, 33(4), 1056–80.

Allahverdi, A. & Al-Anzi, F.S. (2007). Evolutionary heuristics and an algorithm for the two- stage assembly scheduling problem to minimize makespan with setup times, *International Journal of Production Research,* 44(22), 4713–35.

Atashpaz-Gargari, E. & Lucas, C. (2007). Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition, in: *CEC IEEE Congress on Evolutionary Computation*.

Bagheri, A., Zandieh, M., Mahdavi, I. & Yazdani, M. (2010). An artificial immune algorithm for the flexible job-shop scheduling problem, *Future Generation Computer Systems*, 26(4), 533–541.

Castro, L.N.D. & Timmis, J. (2002). An artificial immune network for multimodal function optimization, *Evolutionary computation, CEC'02*, in: Proceedings of the Congress, pp. 699-704.

Cheng, T.C.E. & Wang, G. (1999). Scheduling the fabrication and assembly of components in a two-machine flow shop*, IIE Transactions*, 31, 135–143.

Dasgupta, D. (2002) Special issue on artificial immune system*, IEEE Transactions on Evolutionary Computation*, 6, 225-256.

Demir, Y. & Kursat Isleyen, S. (2013) .Evaluation of mathematical models for flexible job-shop scheduling problems, *Applied Mathematical Modelling*, 37, 977–988.

Dorigo, M. & Gambardella, L.M. (1997). Ant colony system: a cooperative learning approach to the traveling sales man problem, *IEEE Transactions on Evolutionary Computation*, 1(1), 53–66.

Eglese, R.W. (1990). Simulated annealing: a tool for operational research, *European journal of operational research*, 46(3), 271–81.

Fattahi, P., Hosseini, S.M., Jolai, F. & Tavakkoli-Moghaddam, R. (2013). A branch and bound algorithm for hybrid flow shop scheduling problem with setup time and assembly operations*, Applied Mathematical Modelling* 38, 119–134.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence, *Computers and Operations Research*, 13(5), 533–49.

Hariri, A.M.A. & Potts, C.N. (1997). A branch and bound algorithm for the two-stage assembly scheduling problem, *European Journal of Operational Research,* 103, 547–556.

Koulamas, C. & Kyparisis, G.J. (2001). The three-stage assembly flow shop scheduling problem, *Computers and Operations Research,* 28(7), 689–704.

Lee, C.Y., Cheng, T.C.E. & Lin, B.M.T. (1993). Minimizing the makespan in the 3-machine assembly-type flowshop scheduling problem, *Management Science*, 39(5), 616–625.

Luh, G.C. & Chueh, C.H. (2009*).* A multi-modal immune algorithm for the job-shop scheduling problem, *Information Sciences,* 179(10), 1516-1532.

Mladenovic, N. & Hansen, P. (1997). *A variable neighborhood search, Computers & Operations Research*, 24(11), 1097–1100.

Mozdgir, A., Fatemi Ghomi, S.M.T., Jolai, F. & Navaei, J. (2013). Two-stage assembly flow-shop scheduling problem with non-identical assembly machines considering setup times, *International Journal of Production Research*, 51(12), 3625–42.

Potts, C.N., Sevast'janov, S.V., Strusevich, V.A., Wassenhove, L.N.V. & Zwaneveld, C.M. (1995). The two-stage assembly scheduling problem, Complexity and approximation*, Operations Research,* 43(2), 346–355.

Sevkli, M. & Aydin, M.E. (2007). Parallel variable neighbourhood search algorithms for job shop scheduling problems, *IMA Journal of Management Mathematics,* 18(2), 117–133.

Shang, R., Zhanga, W., Li, F., Jiao, L. & Stolkin, R. (2019). Multi-objective artificial immune algorithm for fuzzy clustering based on multiple kernels, *Swarm and Evolutionary Computation*, 50, 100485.

Storn, R. & Price, K. (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous space, *Journal of Global Optimization*, 11(4) 341–359.

Tozkapan, A., Kirca, O. & Chung, C.S. (2003). A branch and bound algorithm to minimize the total weighted flow time for the two-stage assembly scheduling problem, *Computers and Operations Research*, 30(2), 309–20.

Yokoyama, M. (2008). Flow-shop scheduling with setup and assembly operations, *European Journal of Operational Research*, 187, 1184–1195.

Yokoyama, M. (2001). Hybrid flow-shop scheduling with assembly operations, *International Journal of Production Economics,* 73(2), 103–116.

Yokoyama, M. & Santos, D.L. (2005). Three-stage flow-shop scheduling with assembly operations to minimize the weighted sum of product completion times, *European Journal of Operational Research*, 161, 754–770.

Zandieh, M., Fatemi Ghomi, S.M.T. & Moattar Husseini, S.M. (2006). An immune algorithm approach to hybrid flow shops scheduling with sequencedependent setup times, *Applied Mathematics and Computation*, 180(1), 111-127.