چشم‌انداز مدیریت صنعتی
شماره ۲۹ ـ بهار ۱۳۹۷
ص‌ص ۱۱۱ – ۹۳

مدل‌سازی ریاضی و الگوریتم رقابت استعماری برای مسئله خط مونتاژ جریان کارگاهی

حبیب‌رضا غلامی*، اسماعیل مهدی‌زاده، بهمن نادری*****

چکیده

مونتاژ جریان کارگاهی دومرحله‌ای با درنظرگرفتن زمان آماده‌سازی، یکی از مسائل جدید زمان‌بندی تولید است. در این مسئله، قطعات در مرحله نخست در یک سیستم جریان کارگاهی تولید و در مرحله دوم، قطعات تولیدی مونتاژ می‌شوند. هدف از حل این مسئله، ارائه توالی بهینه تولید قطعات و مونتاژ آن‌ها است؛ به‌طوری‌که زمان تکمیل آخرین سفارش کمینه شود. با توجه به جدیدبودن موضوع، تاکنون مدل مناسبی برای حل این مسئله ارائه نشده است. این پژوهش ابتدا به بررسی مدل موجود می‌پردازد و سپس یک مدل برنامه‌ریزی خطی عدد صحیح مختلط برای حل مسئله ارائه می‌دهد؛ سپس برای حل مؤثر این مسئله، دو الگوریتم فراابتکاری رقابت استعماری و ژنتیک ارائه می‌شود. در ادامه عملکرد مدل و الگوریتم‌ها ارزیابی می‌شوند. نتایج نشان می‌دهد الگوریتم رقابت استعماری عملکرد بهتری دارد.

کلیدواژه‌ها: برنامه‌ریزی خطی عدد صحیح مختلط؛ مونتاژ جریان کارگاهی؛ زمان آماده‌سازی؛ الگوریتم رقابت استعماری؛ الگوریتم ژنتیک.

تاریخ دریافت مقاله: ۱۳۹۶/۰۷/۱۸، تاریخ پذیرش مقاله: ۱۳۹۷/۰۳/۲۷.

* دانشجوی دکتری، دانشگاه آزاد اسلامی، قزوین.

** دانشیار، دانشگاه آزاد اسلامی، قزوین (نویسنده مسئول).

Email: emqaiu@yahoo.com

*** دانشیار، دانشگاه خوارزمی.

۱. مقدمه

مدل ریاضی یک نمایش ریاضی از سیستم واقعی است که در برنامه‌ریزی، راهنمای یافتن بهترین نحوه تخصیص منابع محدود و کمیاب است. با توجه به سادگی و کم‌هزینه‌بودن مدل‌های ریاضی و امکان تحلیل سریع آن‌ها، مدل‌های ریاضی در برنامه‌ریزی تولید اهمیت بسزایی دارند. عناصر اصلی یک مدل ریاضی شامل متغیرهای تصمیم‌گیری، محدودیت‌ها و تابع هدف است [۲۱]. مهم‌ترین موارد کاربرد مدل‌های ریاضی، توصیف، پیش‌بینی، ارزیابی گزینه‌ها و آزمون تجربی نظریه‌ها است. مدل‌های ریاضی برحسب معیارهای مختلف به انواع گوناگونی طبقه‌بندی می‌شوند و ساخت هر نوع از این مدل‌ها طی یک فرآیند شش‌مرحله‌ای صورت می‌گیرد که عبارت‌اند از: تعیین متغیرهای تصمیم؛ گردآوری اطلاعات؛ تعیین تجمیع مناسب اطلاعات و طبقه‌بندی آن‌ها؛ تعیین نوع مدل؛ تنظیم و ساخت مدل و ساده‌سازی مدل. پس از مدل‌سازی، ارزیابی و تصحیح آن اهمیت دارد که با توجه به ضوابط چهارگانه دقت، اعتبار، ثبات و سهولت در برآورد متغیرها انجام می‌شود. بعد از آماده‌شدن مدل باید آن را حل کرد که مهم‌ترین روش‌های حل مدل‌ها را می‌توان تحلیل ریاضی، برنامه‌ریزی خطی، برنامه‌ریزی اعداد صحیح و برنامه‌ریزی غیرخطی برشمرد.

مدل‌سازی مسائل زمان‌بندی تولید با زبان ریاضی می‌تواند یکی از نخستین و اصلی‌ترین گام‌های مطالعه باشد. در مدل ریاضی، مشخصه‌های یک مسئله از حالت توصیفی به‌صورت دقیق ریاضی تبدیل می‌شوند؛ ضمن اینکه مدل‌های ریاضی را می‌توان به‌عنوان راه‌حل نیز در نظر گرفت. بیان این نکته که الگوریتم‌های شاخه و کران، شاخه و قیمت و آزادسازی لاگرانژین بر مبنای مدل ریاضی توسعه یافته‌اند، اهمیت مدل ریاضی را روشن می‌سازد؛ بنابراین مدل‌های ریاضی را می‌توان پایه‌ای برای توسعه الگوریتم‌های حل به‌شمار آورد. معمولاً مسائل زمان‌بندی تولید از طریق مدل‌های خطی عدد صحیح ترکیبی یا به‌اختصار MILP فرموله می‌شوند. مدل‌های مختلفی را می‌توان برای یک مسئله از طریق تعریف متغیرهای مختلف بنا کرد. این مدل‌ها می‌توانند توانایی‌های مختلفی برای حل مسئله داشته باشند. نظر به اینکه که مدل‌های مختلفی را می‌توان برای یک مسئله بنا کرد، این بحث مطرح می‌شود که کاراترین مدل چه خواهد بود.

در محیط‌های تولیدی، محیط جریان کارگاهی یکی از پرکاربردترین محیط‌ها در صنعت است [۲۸]. دلیل این ادعا ماهیت بسیاری از محصولات است؛ چراکه در بسیاری از محصولات، شروع یک مرحله جدید از تولید محصول به تکمیل مراحل قبلی آن محصول نیازمند است. افزایش تنوع نیازهای مشتریان از یک‌سو و بازار رقابتی از سوی دیگر، به تولید محصولات پیچیده‌تر منجر شده است. هرچه محصول پیچیده‌تر باشد، یعنی از اجزای بیشتری تشکیل شده است که

امر تعداد مراحل تولید محصول را افزایش می‌دهد. در پایان تمامی اجزای محصول که به‌صورت مجزا یا وابسته به هم تولید شده‌اند، باید در کنار یکدیگر قرار بگیرند تا درنهایت کالای موردنظر تکمیل شود. بیشتر کالاهای تولیدی، حتی اگر ساده باشند، دارای حداقل یک مرحله مونتاژ هستند؛ بنابراین اهمیت مرحله مونتاژ در صنعت امروز بیش از گذشته احساس می‌شود. کاربردهای مسئله جریان کارگاهی مونتاژ توسط پاتس و همکاران (۱۹۹۵) در صنعت تولید کامپیوتر و نوایی و همکاران (۲۰۱۰)، در صنعت تولید مبلمان و الله‌وردی و الانزی (۲۰۰۶)، در توزیع پایگاه داده توضیح داده شده‌اند [۲۰، ۱۹، ۱].

در مرحله نخست، قطعات موردنیاز برای مونتاژ محصولات سفارش‌داده‌شده تولید می‌شود؛ سپس در مرحله دوم قطعات تولیدشده مونتاژ می‌شوند. در مرحله نخست قبل تولید هر قطعه باید ماشین تولیدی آماده‌سازی شود. تنها درصورتی‌که دو قطعه مشابه از دو سفارش مختلف پشت سر هم روی یک ماشین تولید شود، احتیاجی به آماده‌سازی نیست. زمانی می‌توان مونتاژ یک سفارش را شروع کرد که تمامی قطعات آن در مرحله نخست تولید شده باشد.

در این پژوهش ابتدا مدل ریاضی مسئله در قالب یک مدل برنامه‌ریزی عدد صحیح مختلط ارائه می‌شود؛ سپس ازآنجاکه مسئله NP-hard است، برای حل مؤثر آن، دو الگوریتم رقابت استعماری و ژنتیک ارائه می‌شود. برای ارزیابی عملکرد مدل و دو الگوریتم، از آزمایش عددی استفاده شده و نتایج آن توسط ابزارهای آماری تحلیل می‌شود.

در ادامه این مقاله، در بخش دوم، مبانی نظری مسئله جریان کارگاهی و حالت مونتاژی آن بررسی می‌شود. در بخش سوم، مسئله موردمطالعه تعریف و مدل ریاضی آن ارائه می‌شود. در بخش چهارم، دو الگوریتم رقابت استعماری و ژنتیک برای مسئله ارائه می‌شود. در بخش پنجم، با استفاده از دو مجموعه مثال، مدل ریاضی و دو الگوریتم پیشنهادی ارزیابی می‌شوند. درنهایت در بخش ششم، مقاله جمع‌بندی می‌شود.

۲. مبانی نظری و پیشینه پژوهش

نخستین مطالعات در مورد موضوع موردبررسی توسط لی و همکاران (۱۹۹۳) و پاتس و همکاران (۱۹۹۶) با درنظرگرفتن دو مرحله برای آماده‌سازی محصول انجام شد [۱۵، ۲۰]. با بررسی مبانی نظری موضوع مشخص شد که در ابتدا پژوهشگران بسیاری همچون الله‌وردی و ال انزی (۲۰۰۶) و تزکاپان و همکاران (۲۰۰۳)، این مسئله را با تنها یک ماشین در مرحله دوم در نظر گرفتند [۱، ۲۶]. بعد از پاتس و همکاران (۱۹۹۵) چند پژوهش دیگر با توسعه مدل و افزودن محدودیت‌هایی همچون درنظرگرفتن سه مرحله مونتاژ جریان کارگاهی توسط کولاماس و کیپاریسیس (۲۰۰۱)، و حاتمی و همکاران (۲۰۱۰) انجام شد [۲۰، ۱۳، ۸]. رویز و الله‌وردی

(۲۰۰۷)، مسئله اولیه مطرح‌شده را با فرض زمان‌های آماده‌سازی غیروابسته به توالی مدل و حل کردند [۲۲]. بعدها برای واقعی‌ترشدن مسئله هندی‌زاده و همکاران (۲۰۰۷)، زمان‌های آماده‌سازی را وابسته به توالی فرض کردند و حتی این فرض در هر دو فاز تولید و مونتاژ اعمال شد [۹].

توابع هدف مختلفی برای این مسئله در نظر گرفته شده است. ازجمله میکسپن، متوسط زمان تکمیل، کمینه‌سازی دیرکرد و زودکرد، حداقل‌کردن تعداد کارهای دارای دیرکرد و بسیاری دیگر که توسط جوادیان و همکاران (۲۰۰۹) ، سانگ و جان (۲۰۰۹)، و نوایی و همکاران (۲۰۱۴)، بررسی شده‌اند [۱۴، ۲۴، ۱۹]. بیشتر پژوهش‌های صورت‌گرفته با درنظرگرفتن زمان انجام شده‌اند؛ ولی پژوهش‌هایی نیز وجود دارند که در آن‌ها زمان با توجه به تأثیری که بر هزینه‌ها دارد در نظر گرفته شده است. برای مثال، فورست (۱۹۸۳)، را با تابع هدف کمینه‌سازی مجموع هزینه‌های نگهداری و تولید حل کردند [۶].

پژوهش‌های صورت‌گرفتهدر این موضوع را با درنظرگرفتن روش‌های حل نیز می‌توان دسته‌بندی کرد. الله‌وردی و انزی (۲۰۰۶)، از الگوریتم تکاملی برای حل مسئله استفاده کردند [۱]. الگوریتم ژنتیک توسط کارافا و همکاران (۲۰۰۱)، و اسدزاده و زمانی‌فر (۲۰۱۰)، استفاده شده است [۵، ۲]. سوکه و بینگول (۲۰۰۶)، ترکیبی از این الگوریتم‌ها را به‌کار بردند [۲۳]. حریری و پاتس (۱۹۹۷)، با الگوریتم شاخه و کران حد پایینی را برای مسئله ارائه کردند [۷]. تراب‌زاده و زندیه (۲۰۱۰)، برای زمان‌بندی دومرحله‌ای مونتاژ جریان کارگاهی، الگوریتم شبیه‌سازی تبرید را بر اساس تئوری ابرها ارائه کردند [۲۵].

مقاله‌های که اخیراً در این موضوع چاپ شده‌اند، بیشتر بر توسعه مسئله با فرضیه‌های واقعی تمرکز داشته‌اند. برای مثال، کاظمی و همکاران (۲۰۱۷) فرض بسته‌بندی و تحویل را به مسئله اضافه کرده‌اند [۱۱]. وو و همکاران (۲۰۱۸)، اثر یادگیری را در مسئله جریان کارگاهی بررسی کردند [۲۷]. لین و همکاران (۲۰۱۷)، مسئله را در حالت توزیع‌شده مطالعه کردند [۱۶] و کمکی و کیانفر (۲۰۱۵)، مسئله را فرض زمان‌های دردسترس در نظر گرفتند [۱۲].

مسئله موردبررسی در پژوهش حاضر، مسئله جریان کارگاهی مونتاژ مطابق با فرضیه‌های نوایی و همکاران (۲۰۱۴)، برای کاهش مجموع هزینه‌های نگهداری و تأخیر در مسئله مونتاژ جریان کارگاهی با زمان‌های آماده‌سازی وابسته به توالی است [۱۸]. با بررسی مبانی نظری موضوع مشخص شد که تاکنون مدل مناسبی برای این مسئله ارائه نشده است. تنها مدل موجود (نوایی و همکاران (۲۰۱۴)) نیز به‌دلیل وجود ضعف‌های بسیار (تعداد متغیر بالا و غیرخطی‌بودن)، کارایی لازم برای حل مسئله را ندارد و حتی در ابعاد کوچک مسئله نیز، جواب بهینه ارائه نمی‌دهد؛ بنابراین در این پژوهش سعی می‌شودکه برای موضوع مطرح‌شده یک مدل برنامه‌ریزی

خطی عدد صحیح مختلط ارائه شود. این مدل با استفاده از نرم‌افزارهای موجود، در ابعاد کوچک و در زمانی کوتاه قابل‌حل است؛ علاوه بر آن دو الگوریتم رقابت استعماری و ژنتیک برای حل مسئله تطبیق داده می‌شوند.

۳. روش‌شناسی پژوهش

بیان مسئله و مدل‌سازی ریاضی. در این بخش پس از تشریح مسئله، مدل‌های ریاضی آمده است. در این مسئله، n سفارش وجود دارد که هر سفارش شامل g قطعه است. برای تکمیل هر قطعه باید m عملیات تولید انجام شود. بعد دردسترس قرارگرفتن g قطعه، می‌توان عملیات مونتاژ را انجام داد و با مونتاژ این قطعات یک محصول (یکی از سفارش‌ها) را تکمیل کرد. درصورتی‌که دو قطعه مختلف از یک سفارش و یا از سفارش‌های دیگر در توالی تولیدی پست سر هم باشند باید آماده‌سازی انجام شود؛ به‌عبارتی تنها زمانی به آماده‌سازی ماشین برای پردازش احتیاج نیست که یک قطعه یکسان برای دو سفارش مختلف پشت سر هم باشند. در این مسئله فرض بر این است که تمامی قطعات و ماشین‌ها در زمان صفر دردسترس هستند. در این بخش ابتدا نمادهای استفاده‌شده در مدل ریاضی نخست معرفی می‌شوند؛ سپس مدل ریاضی نخست و محدودیت‌های آن موردبررسی قرار می‌گیرند.

اندیس‌های مورداستفاده در مدل نخست و تعریف آن‌ها به‌صورت زیر است:

n : تعداد سفارش‌ها

k : شمارنده سفارش‌ها $\{1, 2, \dots, n\}$

g : تعداد قطعه‌ها

j : شمارنده قطعه‌ها $\{1, 2, \dots, g\}$

m : تعداد ماشین‌ها

i, l : شمارنده ماشین‌ها $\{1, 2, \dots, m\}$

$p_{j,i}$: زمان پردازش قطعه j روی ماشین i

$s_{j,i}$: زمان آماده‌سازی قطعه j روی ماشین i

همچنین متغیرهای تصمیم تعریف‌شده در مدل نخست نیز به‌صورت زیر است.

$X_{k,j,k',j'}$:متغیر باینری که عدد یک می‌گیرد اگر قطعه j در سفارش k بعد از قطعه j’ در سفارش k’ پردازش شود، در غیر این صورت عدد صفر خواهد داشت.

$Y_{k,j,j'}$:متغیر باینری که عدد یک می‌گیرد اگر قطعه j در سفارش k بعد از قطعه j' همان سفارش پردازش شود، در غیر این صورت عدد صفر خواهد داشت.

$C_{k,j,i}$: متغیر پیوسته برای زمان تکمیل قطعه j در سفارش k در ماشین i

F_k :متغیر پیوسته برای زمان تکمیل سفارش k

C_{max} : متغیر پیوسته برای زمان تکمیل آخرین فعالیت

مدل ریاضی نخست به‌صورت زیر است:

Min C_{max}

Subject to:

$$C_{1,j,i} \geq s_{j,i} + p_{j,i} \qquad \forall_{j,i} \qquad (۱)$$

$$C_{k,j,i} \geq C_{k,j,i-1} + p_{j,i} \qquad \forall_{k,j,i>1} \qquad (۲)$$

$$C_{k,j,i} \geq C_{k-1,j,i} + p_{j,i} \qquad \forall_{k>1,j,i} \qquad (۳)$$

$$C_{k,j,i} \geq C_{k',j',i} + s_{j,i} + p_{j,i} - M \cdot \left(1 - X_{k,j,k',j'}\right) \qquad \forall_{k<g,k'>k,j\neq j',i} \qquad (۴)$$

$$C_{k',j',i} \geq C_{k,j,i} + s_{j',i} + p_{j',i} - M \cdot \left(X_{k,j,k',j'}\right) \qquad \forall_{k<g,k'>k,j\neq j',i} \qquad (۵)$$

$$C_{k,j,i} \geq C_{k',j,i} + p_{j,i} - M \cdot \left(1 - X_{k,j,k',j}\right) \qquad \forall_{k<g,k'>k,j,i} \qquad (۶)$$

$$C_{k',j,i} \geq C_{k,j,i} + p_{j,i} - M \cdot \left(X_{k,j,k',j}\right) \qquad \forall_{k<g,k'>k,j,i} \qquad (۷)$$

$$C_{k,j,i} \geq C_{k,j',i} + s_{j,i} + p_{j,i} - M \cdot \left(1 - Y_{k,j,j'}\right) \qquad \forall_{k,j<n,j'>j,i} \qquad (۸)$$

$$C_{k,j',i} \geq C_{k,j,i} + s_{j',i} + p_{j',i} - M \cdot \left(Y_{k,j,j'}\right) \qquad \forall_{k,j<n,j'>j,i} \qquad (۹)$$

$$F_k \geq F_{k-1} + p_k \qquad \forall_{k>1} \qquad (۱۰)$$

$$F_k \geq C_{k,j,m} + p_k \qquad \forall_{k,j} \qquad (۱۱)$$

$$C_{max} \geq F_n \qquad (۱۲)$$

$$C_{k,j,i}, F_k \geq 0 \qquad \forall_{k,j,i} \qquad (۱۳)$$

$$X_{k,j,k',j'} \in \{0,1\} \qquad \forall_{k<g,k'>k,j,j',i} \qquad (۱۴)$$

$$Y_{k,j,j'} \in \{0,1\} \qquad \forall_{k,j>j'} \qquad (۱۵)$$

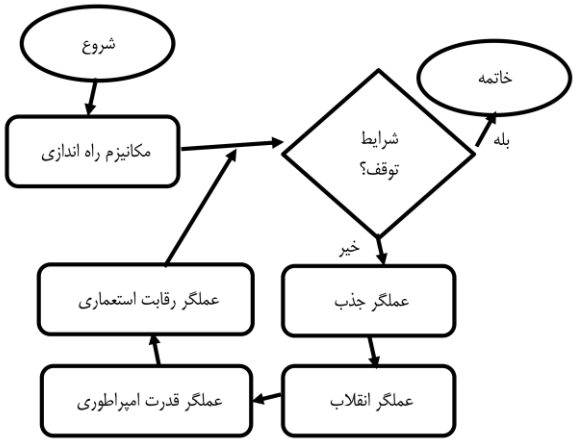
محدودیت ۱، تضمین می‌کند که زمان تکمیل کارها حداقل بزرگ‌تر از مجموع زمان پردازش و آماده‌سازی باشد. محدودیت ۲، تضمین می‌کند که زمان تکمیل هر قطعه حداقل به‌اندازه زمان پردازش بزرگ‌تر از زمان تکمیل همان قطعه در ماشین قبل باشد. محدودیت ۳، نشان می‌دهد

که هر قطعه سفارش‌ها به‌ترتیب تولید شوند. محدودیت‌های ۴، ۵)، ۶ و ۷، زمان تکمیل دو قطعه دو سفارش مختلف را بر اساس توالی تعیین‌شده مرتبط می‌کند ازآنجایی‌که یک ماشین نمی‌تواند هم‌زمان بیش از یک قطعه را پردازش کند. محدودیت‌های (۸) و (۹) زمان تکمیل دو قطعه یک سفارش را بر اساس توالی تعیین‌شده مرتبط می‌کنند. ازآنجاکه یک ماشین نمی‌تواند هم‌زمان بیش از یک قطعه را پردازش کند، محدودیت ۱۰، تضمین می‌کند که سفارش‌ها به‌ترتیب تکمیل شوند. محدودیت ۱۱، زمان تکمیل سفارش‌ها و محدودیت ۱۲، میکسپن را محاسبه می‌کنند. محدودیت ۱۳، متغیرهای پیوسته و محدودیت‌های ۱۴ و ۱۵، نیز متغیرهای پیوسته را تعریف می‌کنند.

٤. تحلیل داده‌ها و یافته‌های پژوهش

الگوریتم‌های حل پیشنهادی. ازآنجاکه مسئله موردبررسی ازجمله مسائل NP-hard است، برای حل آن دو الگوریتم رقابت استعماری و شبیه‌سازی تبرید توسعه داده می‌شود.

الگوریتم رقابت استعماری. الگوریتم رقابت استعماری یک الگوریتم فراابتکاری مبتنی بر جمعیت است. این الگوریتم نخستین بار توسط آتش‌پز و لوکاس (۲۰۰۷) ارائه شد [۳]. پژوهش عطار و همکاران (۲۰۱۱)، ازجمله عملکردهای موفق این الگوریتم است [۴]. معمولاً الگوریتم‌های فراابتکاری از یک پدیده طبیعی الهام می‌گیرند. این الگوریتم علاوه بر توجه به تکامل زیستی انسان و سایر موجودات، به تکامل اجتماعی و تاریخی او به‌عنوان پیچیده‌ترین و موفق‌ترین حالت تکامل، توجه می‌کند. شبه کد الگوریتم رقابت استعماری در شکل ۱، نشان داده شده است.



شکل ۱. فلوچارت الگوریتم رقابت استعماری

مرحله راه‌اندازی اولیه و شرایط توقف. نخستین مرحله الگوریتم شامل نحوه نمایش جواب، تولید جمعیت اولیه و شکل‌دهی امپراطوری‌های اولیه است. نحوه نمایش جواب به‌صورت یک بردار با ng مؤلفه است که n نشان‌دهنده تعداد سفارش‌ها و g تعداد قطعه هر سفارش است. ازآنجاکه باید از هر قطعه، n بار ساخته شود، در رشته هر شماره قطعه، n بار تکرار شده است. قطعات به‌ترتیب از چپ به راست انتخاب و زمان‌بندی می‌شود. برای مرحله مونتاژ نیز، برای نخستین بار که از همه قطعات در مرحله اول تولید انجام شده بود، یک سفارش مونتاژ می‌شود. جمعیت اولیه الگوریتم (تعداد کشورها) رقابت استعماری به‌صورت تصادفی تولید می‌شود. شرط توقف، یک مقدار ثابت زمان پردازش است. این مقدار ثابت برابر است با: nm. در روش نمایش جایگشتی برای این مسئله جواب تا زمانی که در جایگشت به‌صورت کامل رعایت شود جواب همیشه امکان‌پذیر است و هیچ‌یک از عملگرها کامل‌بودن جایگشت را از بین نمی‌بردند.

تابع هزینه هر یک از کشورها محاسبه می‌شود و به تعداد امپراطوری، از بهترین اعضای این جمعیت که دارای کمترین مقدار تابع هزینه هستند، به‌عنوان استعمارگر انتخاب می‌شوند. جمعیت اقیمانده به‌عنوان کشورهای مستعمره هستند که هر یک به یک امپراطوری تعلق دارند. برای تقسیم مستعمرات اولیه در میان استعمارگرها از «انتخاب چرخه رولت» استفاده شده است. برای انجام این کار با داشتن هزینه همه استعمارگرها، هزینه نرمالیزه آن‌ها بر اساس معادله ۱ محاسبه می‌شود.

$$C_n = e^{(-c_n/\max c_i)},\ \ i = 1,2,\dots,nEmp \qquad \text{رابطه (۱)}$$

c_n هزینه nامین استعمارگر است، C_n هزینه نرمالیزه شده‌ی nامین استعمارگر است و $\max c_i$ بیشینه هزینه در بین تمام استعمارگرها است. $nEmp$ تعداد امپراطوری‌ها است. با داشتن هزینه نرمالیزه، قدرت نسبی نرمالیزه هر استعمارگر بر اساس معادله ۲ محاسبه می‌شود و مبنایی برای تقسیم کشورهای مستعمره در میان استعمارگرها است.

$$P_n = \frac{C_n}{\sum_{i=1}^{nEmp} C_i},\quad \sum_{i=1}^{nEmp} P_i = 1 \qquad \text{رابطه (۲)}$$

روش چرخه رولت یکی از معروف‌ترین روش‌های انتخابی است. در این روش ابتدا تمام مقادیر احتمال انتخاب در کنار یکدیگر چیده شده و سپس یک عدد تصادفی در بازه صفر تا یک تولید می‌شود. انتخاب این بازه به این دلیل است که مجموع مقادیر احتمال انتخاب، همیشه برابر با یک خواهد بود. از مقایسه عدد تصادفی با بازه چرخه رولت، شماره استعمارگر متناظر با عدد

تصادفی مشخص می‌شود. از آنجاکه مقدار احتمال هر استعمارگر، بخشی از فضای چرخ رولت را به خود اختصاص داده است، احتمال انتخاب استعمارگرهای شایسته‌تر (دارای تابع هزینه‌ی کمتر) بیشتر خواهد بود. با این روش همه کشورهای مستعمره به استعمارگرها تخصیص داده می‌شوند و با داشتن حالت اولیه تمام امپراطوری‌ها، الگوریتم رقابت استعماری شروع می‌شود. روند تکامل در یک حلقه قرار دارد که تا برآورده‌شدن شرط توقف، ادامه می‌یابد.

عملگر جذب و انقلاب. در هر امپراطوری، کشور استعمارگر به‌منظور افزایش نفوذ خود سعی می‌کند تعداد مستعمره‌هایش را افزایش دهد؛ بنابراین در هر امپراطوری، کشورهای مستعمره به سمت استعمارگر مربوطه حرکت می‌کنند. در الگوریتم ارائه‌شده سیاست جذب به‌صورت زیر تعریف شده است:

برای تغییر در مستعمره بر اساس امپراطور، دو نقطه در مستعمره انتخاب شده و قبل و بعد از این دو نقطه حذف می‌شود؛ سپس مؤلفه‌های که حذف شده‌اند بر اساس ترتیب نسبی آن‌ها در استعمارگر به جواب اضافه می‌شوند. برای درک بهتر رویه جذب، یک مثال در ادامه توضیح داده می‌شود. یک مسئله با ۳ سفارش و ۳ قطعه را در نظر بگیرید. فرض کنید امپراطور و مستعمره به‌صورت زیر هستند.

امپراطور	۱	۱	۲	۳	۱	۲	۲	۳	۳
مستعمره	۱	۲	۳	۱	۱	۲	۳	۲	۳

اگر دو نقطه به‌ترتیب ۲ تا ۵ باشند. جواب جدید ایجاد شده، به‌صورت زیر خواهد بود.

مستعمره	۱	۲	۳	۱	۱	۲	۲	۳	۳

بروز انقلاب، تغییرات ناگهانی در ویژگی‌های اجتماعی سیاسی یک کشور ایجاد می‌کند. در الگوریتم رقابت استعماری، انقلاب با جابه‌جایی تصادفی یک کشور مستعمره به یک موقعیت تصادفی جدید مدل‌سازی می‌شود. انقلاب از دیدگاه الگوریتمی باعث می‌شود که حرکت تکاملی از گیرکردن در دام بهینه موضعی نجات یابد که در بعضی موارد باعث بهبود موقعیت یک کشور می‌شود و آن را به محدوده‌ای که وضعیت بهتری دارد، انتقال می‌دهد. در الگوریتم پیشنهادشده، در هر امپراطوری هر یک از مستعمره‌ها با احتمال مشخص‌شده‌ای، انقلاب خواهند کرد. در این

عملگر، دو کار به تصادف انتخاب شده و جای دو کار با هم جابه‌جا می‌شود. در حین حرکت مستعمرات به سمت کشور استعمارگر و اجرای سیاست انقلاب، ممکن است بعضی از این مستعمرات به موقعیت بهتری نسبت به کشور استعمارگر دست یابند. در این حالت استعمارگر و مستعمره جای خود را با یکدیگر عوض می‌کنند و الگوریتم با کشور استعمارگر در موقعیت جدید ادامه خواهد یافت. در ادامه، این کشور استعمارگر جدید است که شروع به اعمال سیاست همگون‌سازی بر مستعمرات خود می‌کند.

قدرت امپراطوری، رقابت استعماری و حذف امپراطوری‌های ضعیف. قدرت کشور استعمارگر به اضافه درصدی از قدرت کل مستعمرات آن، قدرت کل یک امپراطوری است؛ بنابراین هزینه کل یک امپراطوری از طریق معادله ۳ محاسبه می‌شود:

$$TC_n = c_n + \xi\{mean(c_n^i)\}$$ رابطه (۳)

TC_n، هزینه کل امپراطوری nام و c_n هزینه استعمارگر nام است. $mean(c_n^i)$، میانگین هزینه مستعمره‌های متعلق به امپراطوری nام است. ξ عددی مثبت است. هر امپراطوری که نتواند بر قدرت خود بیفزاید و قدرت رقابت خود از دست بدهد در جریان رقابت‌های استعمارگری حذف خواهد شد. برای مدل‌کردن این واقعیت فرض می‌شود که امپراطوری در حال حذف، ضعیف‌ترین امپراطوری موجود است؛ بنابراین در هر تکرار از الگوریتم یک یا تعدادی از ضعیف‌ترین مستعمرات، ضعیف‌ترین امپراطوری را برداشته و برای تصاحب این مستعمرات رقابتی در میان کلیه امپراطوری‌ها ایجاد می‌شود. مستعمرات یادشدنه لزوماً توسط قوی‌ترین امپراطوری تصاحب نخواهد شد. بلکه امپراطوری‌های قوی‌تر احتمال تصاحب بیشتری دارند. برای مدل‌سازی رقابت میان امپراطوری‌ها برای تصاحب این مستعمره، ابتدا از روی هزینه کل امپراطوری هزینه کل نرمالیزه‌شده آن از طریق معادله ۴ تعیین می‌شود:

$$NTC_n = e^{(-TC_n/\max TC_i)}$$ رابطه (۴)

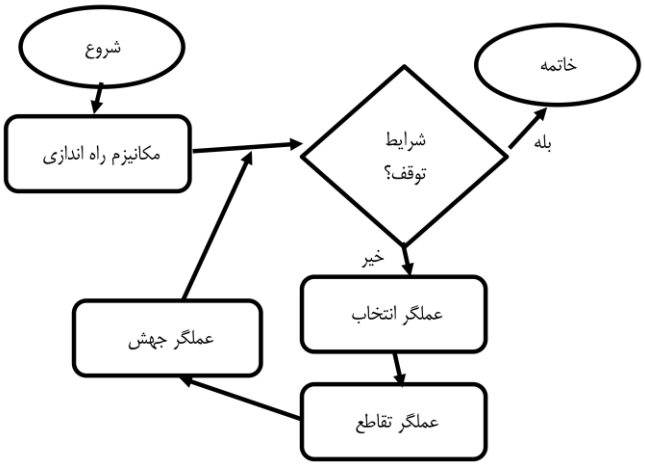
NTC_n هزینه کل نرمالیزه‌شده nامین امپراطوری، TC_nهزینه کل nامین امپراطوری و $\max TC_i$ بیشترین هزینه کل در میان تمام امپراطوری‌ها است (هزینه کل ضعیف‌ترین امپراطوری).

با داشتن هزینه کل نرمالیزه‌شده، احتمال تصاحب مستعمره رقابت توسط هر امپراطوری از طریق معادله ۵ محاسبه می‌شود:

$$pemp_n = NTC_n / \sum_{i=1}^{nEmp} NTC_i$$ رابطه (۵)

با داشتن احتمال تصاحب هر امپراطوری، مستعمره یادشده به امپراطوری که از روش چرخه رولت به‌دست آمده است، تخصیص داده می‌شود. در الگوریتم پیشنهادی، وقتی یک امپراطوری هیچ مستعمره‌ای نداشته باشد به‌عنوان مستعمره به یک امپراطوری دیگر اضافه می‌شود. انتخاب امپراطوری از طریق چرخه رولت صورت می‌پذیرد.

الگوریتم ژنتیک. در این بخش، جزئیات الگوریتم ژنتیک پیشنهادی تشریح می‌شود. الگوریتم ژنتیک یک الگوریتم تصادفی جمعیت‌محور برای حل مسائل بهینه‌سازی ترکیباتی است [۲] شش گام اساسی در طراحی یک الگوریتم ژنتیک برای مسائل بهینه‌سازی شامل طراحی ساختار کرموزوم یا نحوه نمایش حل مسئله، استراتژی تولید جمعیت اولیه، استراتژی انتخاب جمعیت والد یا سازوکار انتخاب، طراحی یا انتخاب عملگرهای ژنتیکی متناسب با ساختار کروموزوم و قیود مسئله، نحوه محاسبه برازندگی یا کیفیت کروموزوم‌ها و تعیین معیار توقف است. فلوچارت الگوریتم ژنتیک در شکل ۲، نشان داده شده است. چگونگی نمایش جواب‌های مسئله، مهم‌ترین بخش در طراحی هر چه بهتر الگوریتم ژنتیک است. ازآنجاکه هر کروموزوم ارائه‌دهنده حلی برای مسئله است، باید قادر باشد ویژگی‌های مسئله را به‌خوبی نشان دهد. از همان روش نمایش توضیح داده‌شده در الگوریتم رقابت استعماری در این الگوریتم نیز استفاده می‌شود.



شکل ۲. فلوچارت الگوریتم ژنتیک

برای تولید جمعیت اولیه عموماً از روش تولید تصادفی کروموزوم‌ها استفاده می‌شود. در روش تصادفی از آنجاکه کروموزوم‌ها متعلق به نواحی مختلف فضای جواب است، تنوع کروموزوم‌ها بالا است؛ درنتیجه، در تکرارهای اولیه الگوریتم، تکامل نسل‌ها سریع‌تر انجام می‌گیرد؛ ولی با افزایش تکرار، تشابه کروموزوم‌ها نیز افزایش می‌یابد تا اینکه در نهایت الگوریتم به یک حل شاخص همگرا شود؛ بنابراین در این پژوهش، جمعیت اولیه در الگوریتم ژنتیک به‌طور تصادفی تولید می‌شود. در الگوریتم ژنتیک، انتخاب جمعیت والد با هدف سوق‌دادن فرآیند جست‌وجو به بخش‌هایی از ناحیه جست‌وجو که امکان یافتن جواب‌هایی با کیفیت بالاتر وجود دارد، انجام می‌گیرد.

در انتخاب بر اساس چرخ رولت، احتمال انتخاب کروموزوم‌ها با برازندگی بیشتر، بالاتر است؛ به‌عبارت‌دیگر به هر کروموزوم به نسبت برازندگی آن، یک شانس انتخاب داده می‌شود؛ درنتیجه ممکن است برخی از کروموزوم‌ها چند بار انتخاب شوند و یا اصلاً انتخاب نشوند؛ بنابراین در این پژوهش، روش چرخ رولت برای انتخاب کروموزوم‌های والد به‌کار می‌رود. این استراتژی به‌منظور تضمین روند صحیح فرآیند جست‌وجو و افزایش عملکرد الگوریتم ژنتیک مورداستفاده قرار می‌گیرد. در این روش برازنده‌ترین اعضای نسل فعلی باید در نسل بعد حضور یابند؛ بدین‌صورت که تعدادی از بهترین کروموزوم‌های جمعیت کنونی (بر مبنای مقادیر برازندگی) بدون تغییر به نسل بعد انتقال می‌یابند. تعداد کروموزوم‌هایی که باید برای استراتژی نخبگی انتخاب شوند برابر با اندازه جمعیت * نرخ استراتژی نخبگی است.

عملگر تقاطع،ترکیب دو والد به امید ایجاد دو اولاد جدید بهتر برای ادامه روند تکامل است. در این پژوهش دو والد توسط سازوکار انتخاب تعیین می‌شوند؛ سپس توسط عملگر تقاطع تک‌نقطه‌ای ترکیب می‌شوند؛ بدین‌صورت که ابتدا یک نقطه در سطر ماتریس ۱ و یا ۲ انتخاب و سطرهای قبل از این نقطه از والد یک به فرزند منتقل می‌شوند؛ سپس سطرهای بعد از این نقطه نیز از والد ۲ به فرزند منتقل می‌شود. تعداد کروموزوم‌هایی که باید برای عملگر تقاطع انتخاب شوند برابر با اندازه جمعیت * نرخ عملگر تقاطع است. عملگر جهش عموماً برای حفظ سطح قابل‌قبولی از تنوع[1] در جمعیت استفاده می‌شود. درواقع عملگر جهش فضایی از جواب را که توسط عملگر تقاطع یافت نشده است، جست‌وجو می‌کند. در این پژوهش، به‌منظور اعمال عملگر جهش، ابتدا سطر و یا ستون از هر یک از دو ماتریس ۱ و یا ۲ انتخاب و اعداد داخل آن به‌تصادف دوباره ایجاد می‌شود.

هنگامی که الگوریتم ژنتیک برای یک مسئله بهینه‌سازی به‌کار می‌رود، با استفاده از تابع هدف مسئله به هر کروموزوم یک مقدار برازندگی نسبت داده می‌شود و با توجه به این مقادیر

1. Diversity

برازندگی، کروموزوم‌ها ارزیابی می‌شوند. الگوریتم ژنتیک زمانی متوقف می‌شود که یک معیار توقف تعیین‌شده ظاهر شود. دو معیار توقف رایج، زمان محاسباتی و یک تعداد معین از نسل هستند. اخیراً، استفاده از زمان محاسباتی به‌عنوان معیار توقف رایج‌تر شده است؛ زیرا مزایایی بیشتری دارد. برای مثال، عملگرهای مختلف، گام‌های محاسباتی مختلفی دارند؛ درنتیجه تخصیص زمان برابر برای پردازش به آن‌ها نسبت به تعداد مراحل برابر عادلانه‌تر خواهد بود. درصورتی‌که زمان حل مسئله از زمان تنظیم‌شده توسط کاربر که تابعی از اندازه مسئله است کمتر باشد، الگوریتم به کار خود ادامه می‌دهد و در غیر این صورت متوقف می‌شود.

ارزیابی عملکرد مدل و الگوریتم‌ها. در این بخش، با استفاده از آزمایش‌های عددی، عملکرد مدل‌های ریاضی و الگوریتم‌های حل ارزیابی می‌شود. در این راستا، دو آزمایش مختلف طراحی می‌شود که آزمایش اول شامل تعدادی مسئله در اندازه کوچک است. مسائل به‌صورت بهینه توسط نرم‌افزار تخصصی بهینه‌سازی حل می‌شوند. آزمایش دوم شامل مسائلی با اندازه بزرگ‌تر است که عملکرد الگوریتم رقابت استعماری را با الگوریتم شبیه‌سازی تبرید مقایسه می‌کند. برای انجام این آزمایش‌ها از نرم‌افزار بهینه‌سازی CPLEX برای حل مدل‌های ریاضی و از نرم‌افزار Matlab برای کدکردن الگوریتم‌های فراابتکاری استفاده می‌شود. از رایانه‌ای با مشخصات Core2Due و ۲ گیگا هرتز حافظه استفاده می‌شود.

مشخصات مسائل در اندازه کوچک شامل تعداد سفارش‌ها ۴، ۶ و ۸، تعداد قطعات ۲، ۳، ۴ و ۵ و تعداد ماشین ۲ در نظر گرفته شده است؛ بنابراین ۱۲ ترکیب از تعداد سفارش‌ها و تعداد قطعات به‌وجود می‌آید. زمان پردازش کارها در هر مرحله به‌طور تصادفی از توزیع یکنواخت در بازه ۱۰ تا ۹۹ تولید می‌شود. برای تولید مسائل در اندازه‌های بزرگ، تعداد سفارش‌ها برابر با ۱۰، ۲۰ و ۵۰ و همچنین تعداد قطعات برابر با ۳ و ۶ لحاظ می‌شود. تعداد ماشین‌ها از یک توزیع یکنواخت بین ۲ تا ۴ تولید می‌شود. برای هر یک از ۶ ترکیب بالا ۱۰ مثال تولید می‌شود که مجموع آن‌ها ۶۰ عدد خواهد شد.

ابتدا مسائلی که با اندازه کوچک تولید شده‌اند توسط الگوریتم رقابت استعماری (ICA) و ژنتیک (GA) حل می‌شود. در ادامه مسائل با اندازه بزرگ توسط هر دو الگوریتم و با تحلیل نتایج عملکرد الگوریتم‌ها موردارزیابی و مقایسه قرار می‌گیرند. قبل از انجام آزمایش‌های مربوط به الگوریتم‌ها، پارامتر هر یک تنظیم می‌شود. الگوریتم رقابت استعماری یک پارامتر *nEmp* دارد. الگوریتم ژنتیک نیز داری پارامتر تعداد افراد *popsize* و درصد تقاطع است. برای هر یک از این پارامترها ۴ سطح در نظر گرفته می‌شود. با حل مثال‌های تولیدشده، درنهایت *nEmp* برابر ۴۰ و *popsize* برابر با ۵۰ و نرخ تقاطع ۶۰ درصد بهترین عملکرد را داشتند.

ابتدا عملکرد این الگوریتم‌ها در مقابل جواب بهینه به‌دست‌آمده توسط مدل‌های ریاضی مقایسه می‌شود که نتایج در جدول ۱، نشان داده شده است. با توجه به نتایج،الگوریتم رقابت استعماری عملکرد بهتری در مقایسه با الگوریتم ژنتیک دارد. الگوریتم رقابت استعماری ۱۰ تا از ۱۲ مثال به‌صورت بهینه حل می‌کند؛ درصورتی‌که الگوریتم ژنتیک ۷ مثال را به‌صورت بهینه حل می‌کند.

جدول ۱. نتایج الگوریتم‌ها در اندازه‌های کوچک

الگوریتم رقابت استعماری	الگوریتم ژنتیک	مدل ریاضی	تعداد قطعه	تعداد سفارش
۳۲۶	۳۲۶	۳۲۶	۲	۴
۳۹۵	۳۹۵	۳۹۵	۳	۴
۴۰۱	۴۰۱	۴۰۱	۴	۴
۴۱۳	۴۱۳	۴۱۳	۵	۴
۵۳۷	۵۴۲	۵۳۷	۲	۶
۸۲۴	۸۲۴	۸۲۴	۳	۶
۶۱۷	۶۵۱	۶۱۷	۴	۶
۵۹۲	۶۰۶	۵۹۲	۵	۶
۴۷۷	۴۷۷	۴۷۷	۲	۸
۸۳۸	۸۳۸	۸۳۸	۳	۸
۹۶۱	۹۷۰	۹۵۵	۴	۸
۱۰۸۳	۱۱۲۷	۱۰۳۶	۵	۸

در ادامه، با استفاده از مثال‌های اندازه بزرگ، عملکرد الگوریتم پیشنهادی رقابت استعماری با الگوریتم ژنتیک مقایسه می‌شود. تمامی مسائل با الگوریتم رقابت استعماری و ژنتیک حل شده است و درصد انحراف نسبی[1] یا RPD آن‌ها به‌دست می‌آید.

$$RPD = 100(Sol - Min)/Min$$

Sol مقدار تابع هدف به‌دست‌آمده توسط الگوریتم در یک مثال و Min کمترین مقدار میکسپن به‌دست‌آمده برای آن مثال است. نتایج در جدول ۲، نشان داده شده است. الگوریتم رقابت استعماری با متوسط RPD برابر با ۰/۵۶درصد، عملکردی بهتری نسبت به الگوریتم ژنتیک با RPD برابر با ۱/۵۶ درصد دارد.

1. Relative Precentage Deviation

با استفاده از ابزار آماری تحلیل واریانس یک‌طرفه[1] بر روی مقادیر درصد انحراف نسبی، نتایج بررسی آماری می‌شوند. با توجه به صفربودن مقدار P نابرابری میانگین الگوریتم‌ها را می‌توان نتیجه گرفت. برای بررسی عملکرد الگوریتم‌ها از میانگین و بازه حداقل تفاوت معنادار (LSD)[2] بر روی مقادیر درصد انحراف نسبی به‌دست‌آمده در حل مسائل تصادفی، استفاده شده است. با توجه به شکل ۳، مشخص است الگوریتم رقابت استعماری عملکرد بهتری نسبت به ژنتیک برای حل مسئله موردبررسی دارد و این تفاوت از نظر آماری معنادار است.

جدول ۲. نتایج RPD الگوریتم‌ها در اندازه‌های بزرگ

الگوریتم رقابت استعماری	الگوریتم ژنتیک	تعداد قطعه	تعداد سفارش
۰/۸	۲/۱	۳	۱۰
۰/۶	۱/۸	۶	۱۰
۰/۶	۱/۶	۳	۳۰
۰/۷	۱/۹	۶	۳۰
۰/۳	۱/۴	۳	۵۰
۰/۴	۱/۱	۶	۵۰
۰/۵۶	۱/۶۵	متوسط	

نکته: اعداد ردیف آخر، متوسط اعداد داخل ستون خود هستند.

برای تحلیل تأثیر مقادیر تعداد کار بر عملکرد الگوریتم‌ها نتایج تحلیل شده است. شکل ۴، میانگین انحراف نسبی به‌دست‌آمده از الگوریتم‌ها را در مقادیر مختلف تعداد کار نشان می‌دهد. الگوریتم رقابت استعماری به‌ازای افزایش تعداد کارها عملکرد بهتر و پایدارتری از خود ارائه می‌دهد. همین‌طور حساسیت عملکرد الگوریتم‌ها نسبت به تعداد ماشین‌ها نیز ارزیابی شد. نتایج تحلیل حساسیت نشان داد که عملکرد الگوریتم‌ها تحت تأثیر تعداد ماشین‌ها نیست؛ درنتیجه از ارائه نمودار مربوطه صرف‌نظر می‌شود.

برای بررسی تأثیر پارامترهای مسئله بر پیچیدگی مدل ریاضی مسئله، تعداد متغیرهای باینری و محدودیت مسئله بر اساس پارامترهای تعداد سفارش (n)، تعداد قطعات (k) و تعداد ماشین (m). تعداد متغیر باینری مدل برابر است با:

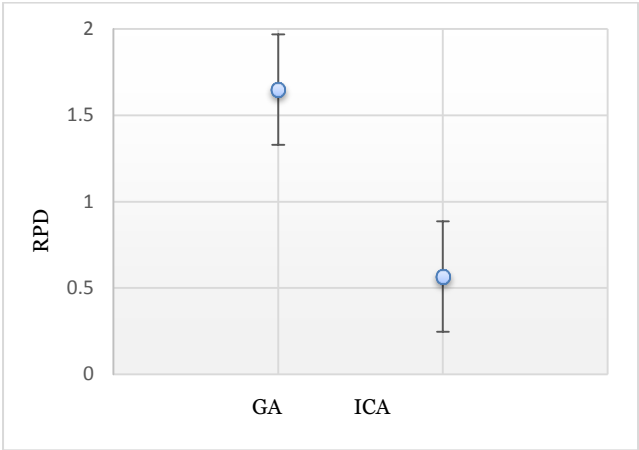
$$n^2g^2 + ng^2$$

1. One-way analysis of variance
2. Least Significant Difference

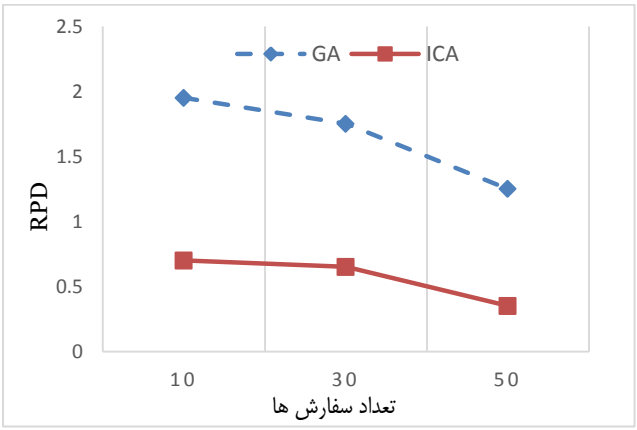
پیچیدگی مدل به تعداد کار و تعداد قطعه درجه دو است و همین‌طور تعداد ماشین در تعداد متغیرها تأثیری ندارد. تعداد محدودیت‌های مدل نیز برابر است با:

$$n^2g^2m + nm(1+g)^2 + g(1+n)$$

حساسیت پیچیدگی مدل به تعداد سفارش و تعداد قطعه، درجه دو است؛ اما حساسیت آن نسبت به تعداد ماشین، درجه یک است؛ درنتیجه تعداد سفارش و تعداد قطعه تأثیر بیشتر بر پیچیدگی مسئله دارند.



شکل ۳. نمایش میانگین و بازه LSD



شکل ۴. نمایش عملکرد الگوریتم‌ها به ازای تعداد کارهای مختلف

۵. نتیجه‌گیری و پیشنهادها

در این پژوهش مسئله زمان‌بندی جریان کارگاهی مونتاژی مطالعه شد. بدین‌صورت که در مرحله نخست، قطعات موردنیاز سفارش‌ها در یک سیستم جریان کارگاهی تولید می‌شود؛ سپس در مرحله دوم، این قطعات مونتاژ می‌شوند. در مرحله نخست، برای تولید قطعات باید پیش از عملیات تولیدی، آماده‌سازی انجام شود. در این پژوهش، یک مدل ریاضی در غالب برنامه‌ریزی عدد صحیح خطی مختلط توسعه داده شد. با استفاده از نرم‌افزارهای تجاری تخصصی تحقیق در عملیات، مدل ریاضی حل شده و نتایج عملکردی آن ارائه شد؛ سپس برای حل مسئله در اندازه‌های واقعی، دو الگوریتم فراابتکاری شامل الگوریتم رقابت استعماری و ژنتیک توسعه داده شد. یک مجموعه مثال آزمایشگاهی تولید و عملکرد الگوریتم‌ها با یکدیگر مقایسه می‌شود. الگوریتم رقابت استعماری در مقایسه با الگوریتم دیگر عملکرد بهتری ارائه کرد.

منابع

1. Allahverdi, A., & Al-Anzi, F.S. (2006). A PSO and a tabu search heuristics for assembly scheduling problem of the two-stage. *Computers & Operations Research*, *33*, 1056-1080.

2. Asadzadeh, L., & Zamanifar, K. (2010). An agent-based parallel approach for the job shop scheduling problem with genetic algorithms. *Mathematical and Computer Modelling*, *52*, 1957-1965.

3. Atashpaz-Gargari, E., & Lucas, C. (2007). Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. *IEEE Congress*, 4661-4670.

4. Attar, S.F., Mohammadi, M., & Tavakkoli-Moghaddam, M. (2011). A novel imperialist competitive algorithm to solve flexible flow shop scheduling problem in order to minimize maximum completion time, *International Journal of Computer Applications*, *28*, 27-32.

5. Caraffa, V., Ianes S.P., Bagchi, T., & Sriskandarajah, C. (2001). Makespan Minimizing makespan in a blocking flowshop using genetic algorithms. *International Journal of Production Economics*, *70(2)*, 101-115.

6. Forst, F. (1983). Minimizing total expected costs in the two-machine, stochastic flow shop. *Operations Research Letters*, *2(2)*, 58-61.

7. 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.

8. Hatami, S., Ebrahimnejad, S., Tavakkoli-Moghaddam R., & Maboudian, Y. (2010). Two meta-heuristics for three stage assembly flowshop scheduling with sequence-dependent setup times. *International Journal of Advanced Manufacturing Technology*, *50*, 1153-1164.

9. Hendizadeh, S.H., ElMekkawy, T.Y., & Wang, G.G. (2007). Bi-criteria scheduling of a flowshop manufacturing cell with sequence dependent setup times. *European Journal of Industrial Engineering*, *1*, 391-413.

10. Khakbiz, M., Rezaei Pendari, A., Dehghan Niri, M., (2017). Mathematical model for diversifying stock portfolio and solving with genetic algorithm, *Industrial Management Prespective*, *24*, 173-196 (In Persian).

11. Kazemi, H., Mahdavi Mazdeh M., & Rostami, M. (2017). The two stage assembly flow-shop scheduling problem with batching and delivery, *Engineering Applications of Artificial Intelligence*, *63*, 98-107.

12. Komaki G.M., Kayvanfar V., (2015). Grey Wolf Optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time, *Journal of Computational Science*, *8*, 109-120.

13. Koulamas, C., & Kyparisis, G.J. (2001). The three-stage assembly flowshop scheduling problem. *Computers & Operations Research*, *28*, 689-704.

14. Javadian, N., Mozdgir, A., Gazani Koohi, E., Davallo Qajar, M.R., & Shiraqai M.E. (2009). Solving assembly flowshop scheduling problem with parallel machines using variable neighborhood search. *In Proceedings of 39th International Conference of Computers and Industrial Engineering, University of Technology of Troyes, Paris, France*, 102-107.

15. 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, 616-625.

16. Lin J.., Wang Z.J., Li X. (2017). A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem, *Swarm and Evolutionary Computation, 36*, 124-135.

17. Navaei J., Ghomi S.M.T.F., Jolai F., Shiraqai M.E., & Hidaji H., (2013). Two-stage flow-shop scheduling problem with non-identical second stage assembly machines. *International Journal of Advanced Manufacturing Technology, 69*, 2215-2226.

18. Navaei, J., Fatemi Ghomi, S.M.T., Jolai, F., & Mozdgir, A. (2014). Heuristics for an assembly flow-shop with non-identical assembly machines and sequence dependent setup times to minimize sum of holding and delay costs. *Computers & Operations Research*, *44*, 52-65.

19. Navaei, J., Mozdgir, A., & Hidaji, H. (2010). Two-stage assembly flow-shop scheduling problem with bi-objective of number of tardy and makespan minimization. *In: Proceedings of the 2010 International Conference on Artificial Intelligence, Las Vegas, Nevada, USA*, 704-709.

20. Potts, C.N., Sevastjanov, S.V., Strusevich, V.A., Van Wassenhove, L.N., & Zwaneveld, C.M. (1995). The two-stage assembly scheduling problem: complexity and approximation, *Operations Research, 43*, 346-355.

21. Rahimi Shekh, H, Sharifi, M, & Shahriari M.R, A model for resialent supply chain: Case study in National healthcare organization, *Industrial Management Prespective, 27*, 127-150 (In Persian).

22. Ruiz, R, & Allahverdi, A. (2007). No-wait flowshop with separate setup times to minimize maximum lateness. *International Journal of Advanced Manufacturing Technology*, *35*, 551-565.

23. Soke, A., & Bingul, Z. (2006). Hybrid genetic algorithm and simulated annealing for two-dimensional nonguillotine rectangular packing problems. *Engineering Applications of Artificial Intelligence, 19*, 557-567.

24. Sung, C.S., & Juhn, J. (2009). Makespan minimization for a 2-stage assembly scheduling problem subject to component available time constraint. *International Journal of Production Economics*, *119*, 392- 401.

25. Torabzadeh, E., & Zandieh. M. (2010). Cloud Theory-Based Simulated Annealing Approach for Scheduling in the Two-Stage Assembly Flowshop. *Advances in Engineering Software, 41*, 1238-1243.

26. Tozkapan, A., Kirca, O., and Chung, C.S. (2003). A branch and bound algorithm to minimize the total weighted flow time for the two-stage assembly scheduling problem, *Computers & Operations Research, 30*, 309-320.

27. Wu C.C., Chen J.Y., Lin W.C., Lai K., Liu S.C., & Yu P.W., (2018). A two-stage three-machine assembly flow shop scheduling with learning consideration to minimize the flowtime by six hybrids of particle swarm optimization, *Swarm and Evolutionary Computation*, doi.org/10.1016/j.swevo.2018.01.012.

28. Zandieh, M., Fotovat, A. (2015). A general flow shop scheduling problem with consideration of position- based learning effect and multiple availability constraints. *Industrial Management Prespective, 20*, 41-58 (In Persian).