نشریه بین المللی مهندسی صنایع و مدیریت تولید زمستان ۱۳۸۹- صفحه ۱۹۰-۱۸۰

شماره ۴، جلد ۲۱، زمستان ۱۳۸۹

نشریه بین المللی مهندسی صنایع و مدیریت تولید

نشریه بین المللی مهندسی صنایع و مدیریت تولید
دانشگاه علم و صنعت ایران

ISSN: 2008-4870

http://IJIEPM.iust.ac.ir/

کمینه‌سازی حداکثر دیرکرد کارها در مسأله زمانبندی جریان کارگاهی جایگشتی دوباره وارد شونده چند ماشینه

مائده فصیحی، فریبرز جولای* و رضا توکلی مقدم

کلمات کلیدی

زمانبندی،
جریان کارگاهی جایگشتی دوباره واردشونده،
حداکثر دیرکرد کارها،
الگوریتم‌های فراابتکاری

چکیده:

در این مقاله، مسأله زمانبندی جریان کارگاهی جایگشتی دوباره وارد شونده با هدف کمینه‌سازی حداکثر دیرکرد کارها مورد بررسی قرار می‌گیرد. محیط جریان کارگاهی دوباره وارد شونده (RFS) همان جریان کارگاهی است با این تفاوت که کارها، ماشین‌های مشخصی را بیش از یک بار ملاقات می‌کنند. در نوع RFS، اگر ترتیب کار روی هر ماشین در هر سطح یکسان باشد، به چنین مسایلی، مسأله جریان کارگاهی جایگشتی دوباره وارد شونده (RPFS) عنوان می‌گردد. در این مقاله، ابتدا مدل ریاضی مسأله کمینه‌سازی حداکثر دیرکرد کارها در RPFS چند ماشینه، توسعه داده می‌شود. برای حل این مسأله، سه الگوریتم فراابتکاری مبتنی بر الگوریتم ژنتیک، شبیه‌سازی تبرید و جستجوی ممنوع طراحی و بکار گرفته می‌شود. الگوریتم‌های فراابتکاری همچنین با حل‌های بهینه ایجاد شده توسط رویکرد برنامه‌ریزی عدد صحیح مقایسه می‌گردند. نتایج آزمایشی نشان می‌دهد که الگوریتم ژنتیک در اکثر موارد کارایی بهتری نسبت به الگوریتم‌های تست شده دیگر دارد.

۱. مقدمه

این تحقیق روی یک مسأله جریان کارگاهی جایگشتی دوباره شونده m-ماشینه با هدف کمینه سازی حداکثر دیرکرد کارها ($T_{\max}$) متمرکز می‌شود. مسأله زمانبندی جریان کارگاهی m-ماشینه بیانگر این است که n کار باید روی m ماشین پردازش شوند و تعدادی عملیات باید در هر کاری انجام شوند. اغلب این عملیات باید روی همه کارها در سفارش یکسان که کارها مسیر یکسانی را طی می کنند، انجام شوند. فرض مسایل زمانبندی کارگاهی کلاسیک که هر کار، هر ماشین را فقط یک بار ملاقات می کند [۱]، اغلب در عمل نقض می‌شود.

تاریخ وصول: ۸۸/۱۲/۲۳

تاریخ تصویب: ۸۹/۷/۱۰

مائده فصیحی، دانش آموخته کارشناسی‌ارشد مهندسی صنایع، دانشکده مهندسی صنایع و مکانیک، دانشگاه آزاد اسلامی واحد قزوین، maede.fasihi@gmail.com

***نویسنده مسئول مقاله: دکتر فریبرز جولای**، دانشیار گروه مهندسی صنایع، پردیس دانشکده‌های فنی، دانشگاه تهران، fjolai@ut.ac.ir

رضا توکلی مقدم، استاد گروه مهندسی صنایع، پردیس دانشکده‌های فنی، دانشگاه تهران، tavakoli@ut.ac.ir

یک نوع جدید کارگاه ساخت، کارگاه دوباره وارد شونده است. خصوصیت اساسی یک کارگاه دوباره وارد شونده این است که یک کار، ماشین‌های مشخصی را بیش از یک بار ملاقات می‌کند [۲]. جریان کارگاهی دوباره وارد شونده (RFS) به این معنی است که n کار وجود دارد که روی m ماشین در یک توالی معین پردازش می شوند و هر کار باید روی ماشین‌ها به صورت $M_1, M_2, ..., M_m, M_1, M_2, ..., M_m, ..., and\ \ M_1, M_2, ..., M_m$ پردازش شود. هر کار می‌تواند به چندین سطح که روی M_1 شروع و در M_m پایان می‌پذیرد تجزیه شود. RFS در بسیاری از سیستم‌های تولیدی وجود دارد، به خصوص در صنایع با تکنولوژی بالا. در ساخت نیمه رساناها، روند پردازش به طور زیادی دوباره واردشونده است به این علت که ویفرها مکرراً یک گروه از تجهیزات را برای اضافه شدن مسیرهای مدارهای متوالی، ملاقات می‌کند [۳] و [۴].

در نوع RFS، اگر ترتیب کار روی هر ماشین در هر سطح یکسان باشد، هیچ کاری مجاز نیست که از کار پیشین عبور کند و به چنین مسایلی، مسأله جریان کارگاهی جایگشتی دوباره وارد شونده (RPFS) عنوان می شود. مسأله زمانبندی جریان کارگاهی

کلاسیک با آزادسازی فرض اینکه هر کار، هر ماشین را بیش از یک بار ملاقات نکند، می تواند به RFS تبدیل گردد. مسأله زمانبندی جریان کارگاهی یکی از مسایل شناخته شده در حوزه زمانبندی است. بیشتر این مسایل به هدف کمینه‌سازی زمان تکمیل برنامه مربوطند. جانسون [۵] در تحقیقات مسایل جریان کارگاهی پیش قدم است. او یک الگوریتم « آسان» برای مسأله جریان کارگاهی دو ماشینه با زمان تکمیل به عنوان مقیاس پیشنهاد کرده است. از آن پس، محققان متعددی روی حل مسایل جریان کارگاهی با m ماشین (m بیش از ۲) تمرکز کردند. اما این‌ها در کلاس NP-hard قرار می گیرند. برای حل این مسایل، روش‌های شمارشی کامل باید به کار گرفته شوند. هنگامی که اندازه مسأله افزایش یابد این روش از نظر محاسباتی، کاربردی نیست. از اینرو محققان دائماً به روش-های ابتکاری توسعه یافته برای مسایل سخت روی می آورند. معیار کارایی می تواند زمان تکمیل کارها، زمان جریان کل کارها، نرخ خروجی و دیرکرد کل در میان دیگر معیارها باشد. سیستم‌های ساخت با خطوط دوباره وارد شونده اخیراً در جامعه علمی بسیار مورد توجه قرار گرفته اند. مسأله دوباره وارد شونده ابتدا در سال ۱۹۸۳ توسط گریوز و همکاران [۶] مورد مطالعه قرار گرفت که در صنعت الکترونیک به کار گرفته می شود. بررسی گوپتا و سیواکومار [۷] یک دید کلی از مطالعات مسایل زمانبندی دوباره وارد شونده را فراهم می‌کند. در ادامه موارد بررسی شده توسط گوپتا و سیواکومار [۷]، مطالعات دقیق و روش‌های تقریبی برای مسأله زمانبندی خط دوباره واردشونده ارایه شده است. اهداف این روش‌ها کمینه‌سازی حداکثر زمان تکمیل ($C_{\max}$) و پس از آن کمینه‌سازی دیرکردهاست. پن و چن [۸] برنامه ریزی عدد صحیح دودوئی توأم را برای مسأله زمانبندی کار کارگاهی توسعه دادند. پن و چن [۹] سه فرمولبندی و برنامه‌ریزی عدد صحیح دودوئی مرکب توسعه یافته و شش روش ابتکاری قابل اجرای توسعه یافته برای حل مسایل زمان‌بندی RPFS جهت کمینه سازی زمان تکمیل برنامه ارایه کردند. چن [۱۰] روش شاخه و کران را برای مسأله زمانبندی جریان کارگاهی جایگشتی دوباره واردشونده با معیار ارزیابی زمان تکمیل برنامه ارایه نموده است. کمینه سازی C_{max} همچنین در روش‌های تقریبی مطالعه شده است. چن و همکاران [۱۱]، [۱۲] و[۱۳] الگوریتم ژنتیک ترکیبی برای مسأله زمانبندی جریان کارگاهی دوباره واردشونده با هدف کمینه‌سازی زمان تکمیل برنامه را بررسی کرده و همچنین الگوریتم‌های جستجوی ممنوع ترکیبی را ارایه نمودند. لیو و وو [۱۴] بهینه‌سازی چند هدفه در فرآیند تولیدی دوباره واردشونده میکرو الکترونیک را مورد مطالعه قرار دادند و از الگوریتم ژنتیک استفاده کرده‌اند و نتایج بررسی‌های آنها نشان داد که الگوریتم ارایه شده، بهبود قابل توجه ای روی تولید میکرو الکترونیک مانند متوسط زمان سیکل، متوسط تعداد کارهای در حال پردازش و نرخ تولید داشته است.

دروبوچویتچ و استروسویچ [۱۵] مسأله زمانبندی با n کار و دو ماشین را در محیط کار کارگاهی دوباره واردشونده با هدف کمینه سازی حداکثر زمان تکمیل را مطالعه کردند و الگوریتمی ابتکاری با کارایی بالا را معرفی نمودند. چئی و کیم [۱۶]، مسأله زمانبندی جریان کارگاهی دوباره واردشونده با m ماشین با هدف کمینه سازی $C_{\max}$ را بررسی و الگوریتم ابتکاری ارایه نمودند. مجموع دیرکرد نهایی در بسیاری از تحقیقات مانند تحقیقات کنگ و همکاران [۱۷]، کمینه شده است. جایی که یک مسأله جریان کارگاهی دوباره وارد شونده با الگوریتمی سه مرحله ای حل شده است. کاری مشابه توسط منج و همکاران [۱۸] انجام شده که الگوریتم ژنتیک برای زیر مسایل مسأله دوباره واردشونده موازی اولیه با آماده سازی و تولید دسته ای بکار گرفته شده است. کوردا و همکاران [۱۹] روشی برای طراحی و کنترل سیستم سلولی جهت دستیابی به کاهش مؤثر زمان دیرکرد تولید در کارگاه های ساخت مقیاس بزرگ با جریان های تولیدی دوباره واردشونده را ارایه کردند و برای حل مسأله، الگوریتم ژنتیک را به کار بردند و مقایساتی برای نشان دادن کارایی روش پیشنهادی خود انجام دادند. چئی و همکاران [۲۰] روی مسأله زمانبندی جریان کارگاهی مختلط با هدف کمینه سازی کل دیرکرد سفارش های دریافت شده تمرکز کرده اند. دمیرکل و ازی [۲۱] یک روش تجزیه را برای کمینه سازی بیشترین دیرکرد برای یک RFS با توالی وابسته به زمان آماده سازی ارایه کردند. الگوریتم شاخه و کران برای مسأله جریان کارگاهی دوباره واردشونده دو ماشینه با هدف کمینه سازی دیرکرد کل توسط چئی و کیم [۲۲] ارایه شد که نتایج حاصل از محاسبات آزمایشی آنها کارایی خوب این الگوریتم را نشان می داد.

از آنجایی که کاهش زمان دیرکرد با توجه به موعد تحویل کالا، از دید مشتری حائز اهمیت است و به نوعی به تولید بهنگام (JIT) اشاره دارد، در این تحقیق زمانبندی RPFS چند ماشینه در محیط قطعی با هدف کمینه سازی حداکثر دیرکرد کارها مورد بررسی قرار گرفته است، با توجه به اینکه در ادبیات چنین بررسی ای تا کنون صورت نگرفته است. این مسأله به طور قطع NP-hard است، زیرا مسأله زمانبندی جریان کارگاهی دو ماشینه با هدف کمینه سازی دیرکرد به عنوان یک NP-hard شناخته شده است [۲۳]. بنابراین چند فرابتکاری مشهور که برای حل مسایل جریان کارگاهی کلاسیک به کار گرفته می‌شوند، جهت کمینه سازی هدف مورد نظر در مسأله مورد بررسی، ارایه شده و با یکدیگر و همچنین برای مسایل کوچک با حل های بهینه ایجاد شده توسط رویکرد برنامه ریزی عدد صحیح (IP)[1] مقایسه می گردند.

این مقاله بدین صورت سازمان دهی شده است: مسأله مورد مطالعه در بخش ۲ شرح داده می شود. در بخش ۳ مدل ریاضی مسأله معرفی می گردد. سه الگوریتم فرابتکاری موجود در ادبیات برای

[1] *Integer Programming*

مسأله مورد نظر در بخش ۴ ارایه می گردد. بخش ۵ نتایج محاسباتی را برای سه الگوریتم نشان می دهد. نتیجه گیری در آخرین بخش آورده می شود.

۲. شرح مسأله و مفروضات

یک مسأله جریان کارگاهی (جایگشتی) فرض می کند که همه عملیات هر کار، هر ماشین را دقیقاً یک بار با ترتیب $M_1,M_2,...,M_m$ ملاقات می کنند. این ترتیب پردازش به عنوان یک سطح تعریف می شود و بنابراین لازمه ی مسیریابی یک کار در RPFS می‌تواند تجزیه شدن به چندین سطح باشد. پس یک جریان کارگاهی جایگشتی کلاسیک یک نوع خاص از RPFS با یک سطح است و بعضی از فرمول های آن می تواند برای حل RPFS عمومی توسعه یابد. پردازش یک کار روی یک ماشین، یک عملیات و مدت مورد نیاز، زمان پردازش نامیده می شود.

مفروضات در نظر گرفته شده برای مسایل زمانبندی RPFS بدین صورت می باشد: هر کاری ممکن است ماشین های مشخصی را بیش از یک بار ملاقات کند. ترتیب ماشین برای هر n کار، یکسان است.

ترتیب کار برای هر m ماشین در هر سطح یکسان است. هر دو عملیات متوالی یک کار باید روی ماشین های متفاوت پردازش شوند. زمان های پردازش مستقل از توالی و قطعیند. همه اطلاعات شناخته و ثابت شده اند. همه ی کارها برای پردازش در زمان صفر که ماشین ها بیکارند و بی واسطه برای کار در دسترسند، آماده اند. ابداً بریدگی جایز نیست. وقتی که یک عملیات شروع می شود، باید قبل از عملیات دیگری که می تواند روی آن ماشین شروع شود کامل شود. ماشین ها هرگز خراب نمی شوند و در تمام مدت زمانبندی در دسترس اند. محدودیت های تکنولوژیکی در پیشرفت و تغییرناپذیری شناخته شده اند. تنها یکی از هر نوع ماشین وجود دارد. فضای انتظار نامحدود برای انتظار کارها برای انجام شدن وجود دارد. نمادهایی که در در این قسمت بکار گرفته شده، بدین صورت می باشند:

n : تعداد کل کارها برای پردازش در زمان صفر / m : تعداد کل ماشین ها در کارگاه / L : تعداد کل سطح های هر کار / J_i : کار شماره i ، $1 \le i \le n$ / M_j : ماشین شماره j ، $1 \le j \le m$ / O_{lj}^i : عملیات کار J_i روی M_j در سطح l ، $1 \le i \le n$، $1 \le l \le L$، $1 \le j \le m$ / P_{lj}^i : زمان پردازش O_{lj}^i ، $1 \le i \le n$ ، $1 \le l \le L$، $1 \le j \le m$ / o : شماره عملیات، $1 \le o \le (m \times L)$

تجزیه سطح می تواند بدین صورت شرح داده شود: فرض کنید کار J_i شامل ۶ عملیات برای پردازش روی ۲ ماشین است که (i,o,j) نشان دهنده این است که عملیات o از کار i باید روی ماشین M_j پردازش شود و مسیر پردازش آن به صورت زیر است:

$$(i,1,1) \to (i,2,2) \to (i,3,1) \to (i,4,2) \to (i,5,1) \to (i,6,2)$$

که هر عملیات زمان پردازش مربوط به خود را داراست که بدین ترتیبند: ۷،۳، ۹، ۲، ۱ و ۵. پردازش J_i می تواند به ۳ سطح تجزیه شود: اولین سطح: $(i,1,1) \to (i,2,2)$، دومین سطح: $(i,3,1) \to (i,4,2)$ و سومین سطح: $(i,5,1) \to (i,6,2)$.

O_{lj}^i را عملیات J_i روی M_j در سطح l قرار داده و P_{lj}^i را زمان پردازش O_{lj}^i در نظر می گیریم. همچنین، L تعداد سطحهای J_i است.

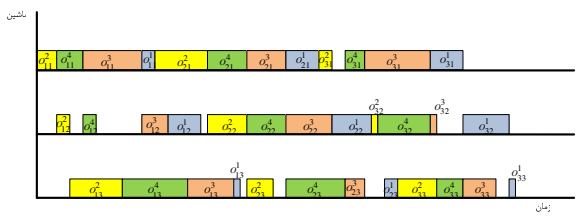
در نتیجه: $L=3$ ، $O_{11}^i=(i,1,1)$ ، $O_{12}^i=(i,2,2)$ ، $O_{21}^i=(i,3,1)$ ، $O_{22}^i=(i,4,2)$ ، $O_{31}^i=(i,5,1)$ ، $O_{32}^i=(i,6,2)$ ، $O_{11}^i=(i,1,1)$ ، $P_{11}^i=7$ ، $P_{12}^i=3$ ، $P_{21}^i=9$ ، $P_{22}^i=2$ ، $P_{31}^i=1$ ، $P_{31}^i=1$ ، $P_{32}^i=5$.

مثال زیر نمایش حل زمانبندی در مسأله RPFS را شرح می دهد. یک مسأله زمانبندی RPFS با ۴ کار، ۳ ماشین و ۳ سطح را در نظر گرفته شده که در جدول ۱ زمان های پردازش برای عملیات هر کار شرح داده شده است.

جدول ۱. شرح زمان های پردازش مثال

کار	سطح ۱	سطح ۲	سطح ۳
	$P_{11}^1=2$	$P_{21}^1=5$	$P_{31}^1=5$
J_1	$P_{12}^1=5$	$P_{22}^1=6$	$P_{32}^1=7$
	$P_{13}^1=1$	$P_{23}^1=2$	$P_{33}^1=1$
	$P_{11}^2=3$	$P_{21}^2=8$	$P_{31}^2=2$
J_2	$P_{12}^2=2$	$P_{22}^2=6$	$P_{32}^2=1$
	$P_{13}^2=8$	$P_{23}^2=4$	$P_{33}^2=6$
	$P_{11}^3=9$	$P_{21}^3=6$	$P_{31}^3=10$
J_3	$P_{12}^3=4$	$P_{22}^3=7$	$P_{32}^3=1$
	$P_{13}^3=7$	$P_{23}^3=3$	$P_{33}^3=5$
	$P_{11}^4=4$	$P_{21}^4=6$	$P_{31}^4=3$
J_4	$P_{12}^4=2$	$P_{22}^4=6$	$P_{32}^4=8$
	$P_{13}^4=10$	$P_{23}^4=8$	$P_{33}^4=4$

شکل ۱ یک حل شدنی (زمانبندی) برای مسأله RPFS را به نمایش می گذارد. همانطور که مشاهده می نمایید، ترتیب ماشین برای هر ۴ کار یکسان است. همچنین ترتیب کار برای هر ۳ ماشین در هر سطح نیز یکسان می باشد. در این زمانبندی، ترتیب کارها برای پردازش روی هر ماشین در هر سطح بدین صورت می باشد: ۲، ۴، ۳ و ۱.



شکل ۱. زمانبندی شدنی برای مثال

در مسایل RPFS، باید روی کارها تمرکز کنیم و روش رمزگذاری بر پایه کار است. علت اصلی این است که وقتی یک بار توالی پردازش مشخص شود، هر ماشین ترتیب مشابه را برای همه کارها پی بگیرد. زمانی که روی کارها به جای عملیات تمرکز شود، مسأله ساده تر خواهد شد. حل های همسایه با تغییر ترتیب کار حل اولیه تولید می شوند [۱۳]. گرچه زمانبندی های جایگشتی در مسأله جریان کارگاهی ۲-ماشینه عمومی با هر معیار کارایی عادی، بهترین حل ها را می دهند [۱]، این در مورد مسأله جریان کارگاهی دوباره وارد شونده صدق نمی‌کند. زمانبندی جایگشتی در مسأله جریان کارگاهی دوباره وارد شونده m-ماشینه با هدف کمینه سازی دیرکرد کارها دارای بهترین حل نیست. زمانبندی جایگشتی حتی برای مسایل جریان کارگاهی دوباره وارد شونده ۲-ماشینه هم بهترین حل را نمی دهد، زیرا ممکن است یک مورد زمانبندی غیر جایگشتی که بهتر از زمانبندی جایگشتی باشد، پیدا شود [۲۲]. در این مقاله، تنها زمانبندی جایگشتی را مورد بررسی قرار می دهیم. اگرچه حل بهینه ممکن است بدست نیاید. زیرا در بیشتر سیستم های واقعی، زمانبندی های جایگشتی به خاطر اجرای آسان یا به خاطر مدیریت جریان مواد، ترجیح داده می شوند. علاوه بر این زمانبندی های غیر جایگشتی در بسیاری از موارد به علت کافی نبودن فضای انبار میانی بین ماشین‌ها و وجود محدودیت های فنی سیستم های حمل مواد، ممکن است قابل اجرا نباشند.

۳. مدل ریاضی مسأله

RPFS توسعه ی جریان کارگاهی کلاسیک است با پذیرفتن اینکه یک کار ماشین های مشخصی را بیش از یک بار ملاقات کند. در این قسمت فرمولبندی مدل RPFS را شرح می دهیم که توسعه یافته دومین مدل پن و چن [۹] می باشد. نمادها در این مدل بدین صورتند (n، m و L قبلاً شرح داده شد):

p_{lk}^{i}: زمان پردازش عملیات کار i روی ماشین k در سطح l

x_{ij}: ۱، اگر کار i در مکان j ام در هر سطح زمانبندی شود؛ در غیر اینصورت صفر

h_{klj}: زمان شروع عملیات زمانبندی شده در مکان j ام سطح l روی ماشین k

C_j: زمان تکمیل کار زمانبندی شده در مکان j ام سطح L روی ماشین m

d_j: موعد تحویل کار زمانبندی شده در مکان j ام

T_j: زمان دیرکرد کار زمانبندی شده در مکان j ام

$T_{\max}$: حداکثر زمان دیرکرد کارها

$$Minimize \quad T_{\max} \tag{۱}$$

Subject to

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, 2, \ldots, n \tag{۲}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, 2, \ldots, n \tag{۳}$$

$$h_{111} = 0 \tag{۴}$$

$$h_{1,1,j+1} = h_{11j} + \sum_{i=1}^{n} x_{ij} p_{11}^{i}, \quad j = 1, 2, \ldots, n-1 \tag{۵}$$

$$h_{1,l,j+1} \geq h_{1lj} + \sum_{i=1}^{n} x_{ij} p_{l1}^{i}, \; l = 2,3,\ldots,L, \; j = 1,2,\ldots,n-1 \tag{۶}$$

$$h_{1,l+1,1} \geq h_{1ln} + \sum_{i=1}^{n} x_{in} p_{l1}^{i}, \quad l = 1, 2, \ldots, L-1 \tag{۷}$$

$$h_{1,l+1,j} \geq h_{mlj} + \sum_{i=1}^{n} x_{ij} p_{lm}^{i}, \; l = 1,2,\ldots,L-1, \; j = 1,2,\ldots,n \tag{۸}$$

$$h_{k,l,j+1} \geq h_{klj} + \sum_{i=1}^{n} x_{ij} p_{lk}^{i},$$
$$k = 1,2,\ldots,m, \; l = 1,2,\ldots,L, \; j = 1,2,\ldots,n-1 \tag{۹}$$

$$h_{k,l+1,1} \geq h_{kln} + \sum_{i=1}^{n} x_{in} p_{lk}^{i}, k = 2,3,\ldots,m, l = 1,2,\ldots,L-1 \tag{۱۰}$$

$$h_{k+1,1,1} = h_{k11} + \sum_{i=1}^{n} x_{i1} p_{1k}^{i}, \quad k = 1, 2, \ldots, m-1 \tag{۱۱}$$

$$h_{k+1,l,j} \geq h_{klj} + \sum_{i=1}^{n} x_{ij} p_{lk}^{i},$$
$$k = 1,2,\ldots,m-1, \; l = 1,2,\ldots,L, \; j = 1,2,\ldots,n \tag{۱۲}$$

$$h_{k+1,l+1,1} \geq h_{kln} + \sum_{i=1}^{n} x_{in} p_{lk}^{i},$$
$$k = 1, 2, \ldots, m-1, \; l = 1, 2, \ldots, L-1 \tag{۱۳}$$

$$C_j = h_{mLj} + \sum_{i=1}^{n} x_{ij} p_{Lm}^{i}, \; j = 1, 2, \ldots, n \tag{۱۴}$$

$$T_j = \max\{0, C_j - d_j\}, \quad j = 1, 2, \ldots, n \qquad (۱۵)$$

$$T_{\max} = \max_j\{T_j\}, \quad j = 1, 2, \ldots, n \qquad (۱۶)$$

$$C_j \geq 0,\ T_j \geq 0,\ h_{klj} \geq 0,$$
$$k = 1, 2, \ldots, m,\ l = 1, 2, \ldots, L,\ j = 1, 2, \ldots, n,$$
$$x_{ij} = 0 \text{ or } 1,\ i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, n \qquad (۱۷)$$

محدودیت (۱) تابع هدف را توصیف می کند. محدودیت های (۲) تا (۵) و (۱۱) در اصل تعریفی هستند، در حالیکه محدودیت های (۶) تا (۱۰)، (۱۲) و (۱۳) بر رابطه اولویت تأکید دارند. محدودیت (۱۴)، C_j را به عنوان زمان پایان پردازش کار زمانبندی شده در مکان j ام، روی ماشین m در آخرین سطح تعریف می کند. قیود نامنفی و دودوئی به ترتیب h_{klj} و x_{ij} در (۱۷) مشخص شده اند.

۴. الگوریتم های فراابتکاری

همانطور که قبلاً ذکر شد مسأله RPFS مورد بررسی در این تحقیق، قویاً NP-hard بوده و برای حل آن از الگوریتم های فراابتکاری کارا بهره می جوییم. در ادامه سه الگوریتم فراابتکاری مشهور به طور خلاصه معرفی شده و برای مسأله مورد بحث بکار گرفته می شوند.

۴-۱. الگوریتم ژنتیک

الگوریتم ژنتیک (GA) یکی از روش های جستجوی فراابتکاری است که هلند [۲۴] ابتدا آن را معرفی کرد. این الگوریتم در کلاس الگوریتم های بهینه سازی تصادفی قرار دارد و بخصوص برای بهینه سازی مسایل پیچیده با فضای جستجوی ناشناخته مناسب است.

ایده اصلی GA از نظریه تکاملی داروین گرفته شده است. به این معنی است که یک والدین خوب فرزند بهتری را ایجاد می کند. GA یک فضای مسأله را با یک جمعیت از کروموزوم ها جستجو می کند و کروموزوم ها را برای تحقیق بعدی با توجه به کارایی آنها انتخاب می نماید.

هر کروموزوم به شکل یک حل در فضای مسأله با مفهوم مسایل بهینه سازی رمزگشایی می شود. عملگرهای ژنتیک، ساختارهای با کارایی بالا (والدین) را در جهت تولید ساختارهای جدیدی که بالقوه مناسبترند (فرزندان)، به کار گرفته اند. بنابراین مجری های خوب، جمعیت را از یک نسل به نسل بعدی انتقال می‌دهند. شبه کد الگوریتم ژنتیک به طور عمومی به صورت زیر می‌باشد:

Algorithm (1): Genetic Algorithm
1: Initialization
1.1: Parameter Setting (Pc, Pm, StopCriteria, PopSize, Selection Strategy, Crossover Op., Mutation Op., Perform. Scalability, NumGen)
1.2: Initialize Population(Randomly)
2: Fitness Evaluation
3: **Repeat**
4: Individual Selection for Mating Pool
(Size of Mating Pool =PopSize)
5: For each consecutive pair apply Crossover
(For each consecutive pair apply Crossover with probability pc)
6: Mutate Childeren
(For each new-born apply mutation with probability pm)
7: Replace the Current Population by the the resulting Mating Pool
8: Fitness Evaluation
9: **Until** Stopping Criteria is met

تعیین پارامترها: پارامترها در GA شامل اندازه جمعیت، تعداد نسل ها، احتمال تقاطع، احتمال جهش، احتمال تولید مجدد و احتمال پردازش دیگر عملگرهای GA می باشد.

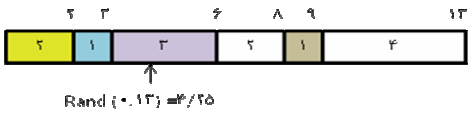
جمعیت اولیه: به طور تصادفی تولید می گردد.

تابع برازندگی: با تابع برازندگی که گاهاً به آن تابع ارزیابی گفته می شود، هر کروموزوم رمزگشایی شده و به آن یک معیار و مقدار برازندگی نسبت داده می شود و در روند انتخاب، کروموزوم های متناسب تر احتمال بیشتری برای انتخاب شدن برای نسل بعدی دارند. برای تعیین تابع برازندگی ابتدا $T_{\max}$ برای همه ی کروموزوم های جمعیت محاسبه می شود و بزرگترین $T_{\max}$ در میان همه کروموزوم ها در جمعیت متداول پیدا شده و به عنوان $f_{\max}$ مشخص می گردد. تفاوت بین هر دیرکرد (f_i) و $f_{\max}$ با توان ۱/۰۰۵ مقدار برازندگی آن کروموزوم مشخص است. مقیاس پایین توان (α) توسط گیلیز [۲۵] پیشنهاد داده شد. عموماً این مقدار وابسته به مسأله است. گیلیز مقدار ۱/۰۰۵ را گزارش داد. تابع برازندگی با $F_i = (f_{\max} - f_i)^{\alpha}$ مشخص شده است، برای تضمین اینکه احتمال انتخاب برای زمانبندی با زمان دیرکرد پایین تر، بالا باشد.

انتخاب: فرآیندی برای انتخاب فرزند از والدین برای نسل بعدی است. طبق این تعریف عمومی انتخاب تصادفی کروموزوم باید درجه کارایی کروموزوم در جمعیت را نشان دهد. بنابراین یک والد با کارایی بالاتر شانس بیشتری برای انتخاب شدن برای نسل بعدی دارد. در این تحقیق فرآیند انتخاب والدین از طریق فرآیند انتخاب چرخه رولت به کار گرفته می شود که در زیر شرح داده شده است.

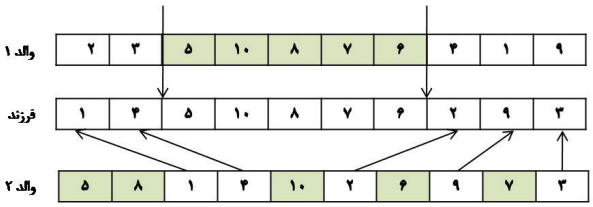
- محاسبه مقدار برازش برای هر کروموزوم از جمعیت (کروموزوم های با برازش بالاتر، شانس بیشتری برای انتخاب دارند).

- محاسبه برازش تجمعی برای هر کروموزوم.
- فرض: برازش تجمعی آخرین کروموزوم Sum باشد.
- تولید عددی تصادفی در فاصله (Sum , 0).
- مقایسه عدد مربوطه با برازش های تجمعی و انتخاب کروموزومی که در فاصله مربوطه قرار گرفته.



شکل۲. مثال برای انتخاب چرخه رولت

تقاطع: تقاطع عملیات تولید یک رشته ی جدید (فرزند) از دو رشته والد است. این عملگر اصلی GA است. موراتا [۲۶] نشان داد که تقاطع دو نقطه ای برای مسایل جریان کارگاهی مؤثر است. از این رو روش تقاطع دو نقطه ای در این تحقیق به کار گرفته می شود. تقاطع دو نقطه ای در شکل ۳ نشان داده شده است.



شکل۳. تقاطع دونقطه ای

جهش: جهش عملگر کاربردی دیگر GA است. چنین عملیاتی می-تواند به عنوان انتقال از یک حل متداول به حل همسایه اش در یک الگوریتم تحقیق محلی دیده شود. بکار گیری این عملگر از بهینگی زودرس و افتادن در بهینه محلی جلوگیری می کند.

تولید مجدد: این عملگر با توجه به احتمال تخصیص یافته به آن، شماری از بهترین حل ها را به نسل بعدی انتقال می دهد.

توقف: شرط های مورد استفاده ی معمول برای توقف پروسه ی GA عبارتند از (۱) تعداد نسل (۲) یک هدف مشخص (۳) همگنی جمعیت.

۲-۴. الگوریتم شبیه سازی تبرید

تکنیک شبیه سازی تبرید (SA) پیشنهاد شده توسط کریک پاتریک و همکاران [۲۷] یک روش جستجوی تکرار شونده، تصادفی و بر پایه همسایگی است که ایده اصلی این الگوریتم از تناسب بین رویه فیزیکی ذوب فلزات و نحوه سرد شده آنها و استراتژی حل مسایل بهینه سازی ترکیبیاتی گرفته شده است. شبه SA کد بدین صورت است:

Algorithm (2): Simulated Annealing
1: **input:** an instance x of a CO problem
2: $S \leftarrow$ Generate *initial solution*
3: $T \leftarrow$ Set *initial temperature*
4: **while** termination conditions not met **do**
5: $S' \leftarrow$ Pick neighbor at random($N(S)$)
6: **if** $f(S') \leq f(S)$ **then**
7: $S \leftarrow S'$;
8: **else**
9: accept S' as new solution with probability $p(T, S', S)$
10: **end if**
11: Adapt *temperature*(T)
12: **end while**
13: $Sbest \leftarrow S$
14: **output:** *Sbest*, "candidate" to optimal solution for x

این روش از یک جواب اولیه همچون π_0 آغاز کرده، سپس بصورت متوالی در بین همسایگی ها با توجه به ساختار همسایگی های از پیش تعیین شده حرکت می نماید تا به شرایط توقف الگوریتم رسیده و متوقف شود. این الگوریتم در هر تکرار یک جواب تصادفی مانند π_0 از همسایگی جواب فعلی (π') انتخاب کرده و این حل جدید (π_i) را با احتمال زیر به عنوان جواب بعدی می‌پذیرد.

$$P(T_i,\pi_i,\pi') = \min\left\{1, \exp(-\frac{F(\pi') - F(\pi_i)}{T_i})\right\} \qquad (۱۸)$$

در این رابطه T_i مقدار دما در تکرار فعلی است. این دما در روند الگوریتم سیر نزولی داشته و این سرد شدن تدریجی با توجه به الگوی خاصی صورت می گیرد.

حل اولیه: به طور تصادفی ایجاد می گردد.

دمای ابتدایی: با توجه به مسأله و تعداد مراحلی که لازم است به دمای پایانی برسد، مقدار می گیرد.

الگوی تغییر دما: تغییر دما در هر تکرار از الگوریتم با توجه به رابطه زیر صورت می گیرد:

$$T_i = T_0 - \left(\frac{T_0 - T_f}{N}\right) \times i \qquad (۱۹)$$

به طوریکه T_i دما در مرحله i ام و N تعداد مراحلی است که باید از T_0 به T_f برسیم.

دمای پایانی: دمای پایانی صفر در نظر گرفته شده است.

تعیین همسایه: در اینجا روش تعویض تصادفی دو کار را برای تولید همسایگی انتخاب می‌کنیم.

۳-۴. الگویتم جستجوی ممنوع

جستجوی ممنوع (TS) که توسط گلور [۲۸] معرفی شده، یک فراابتکاریست که رویه جستجوی ابتکاری محلی را برای پیدا کردن فضای حل برتر از بهینگی محلی پی می گیرد. پروسه محلی یک جستجو است که عملیات حرکت برای تعیین همسایه هر حل در دست را به کار می گیرد. یکی از اجزا اصلی TS، کاربرد حافظه انطباقی است که رفتار جستجوی انعطاف پذیرتر را ایجاد می کند.

طریقه عمل TS بدین صورت است: شروع از یک حل شدنی اولیه، در هر گام ما یک حرکت به سمت یک حل همسایه را انتخاب می کنیم که در این روش به سمت یک حل که امیدواریم کمترین مقدار تابع هدف را دارا باشد، حرکت می کنیم. شکل ابتدایی الگوریتم TS به صورت زیر است:

Algorithm (3): Tabu Search
1: **Set** TabuList={} ; TabuL=TL ; k=1;
2: Choose an initial solution $s \in S$;
3: ŝ=s;
4: **Repeat**
5: Generate N(s,k) $\subseteq$ N(s);
6: Evaluate each s$\in$ N(s,k);
7: Modify the neighborhood N*(s,k)=N(s,k) $\ominus$TabuList;
8: Choose the best solution s´ $\in$ N*(s,k);
9: Move to s=s´;
10: **If** solution s is better than ŝ **then** ŝ=s;
11: Update the TabuList;
12: k=k+1;
13: **Until** stopping condition is met

حل اولیه: در این مقاله، حل اولیه به صورت تصادفی تولید می‌گردد.

جستجوی همسایگی: جستجوی همسایگی از حل در جریان شروع می گردد و تا پیدا کردن حل شدنی و شاید بهتر در همسایگی اش پیگیری می شود. اگر حل همسایه بهتر از حل در جریان باشد، حل در جریان با حل همسایه جایگزین می شود تا زمانی که شرط های توقف ارضا شوند. حل های همسایه با تغییر ترتیب کار حل اولیه تولید می شوند. روش تعویض جفتی را برای تولید همسایگی در اینجا برمی گزینیم.

انتخاب یک حرکت: در ابتدا، حداکثر زمان دیرکرد ($T_{\max}$) برای هر حل همسایه محاسبه می شود. بعد حلی که کمترین $T_{\max}$ را در بین دیگر حل ها دارد و خارج از لیست ممنوع است، به عنوان یک حرکت انتخاب می شوند.

ثبت بهترین حل تا اینجا: اگر حل بعد از حرکت بهتر از بهترین حل تا اینجا باشد، بهترین حل تا اینجا با حل بعد از حرکت جایگزین می شود و لیست ممنوع را به هنگام می نماییم.

ثبت در لیست ممنوع: ثبت شماره کارهایی که تعویض شده اند را در لیست ممنوع بکار می گیریم. پس با این کار، خواه دو کار معاوضه شده اند یا نه می توانند در لیست ممنوع نگه داشته شوند، حرکت $v=(x,y)$ به لیست ممنوع T با روش استاندارد زیر اضافه می شوند. لیست ممنوع T، یک موقعیت را به سمت جلو انتقال می‌دهد و v را در آخرین موقعیت در لیست قرار می دهد: $T_j = T_j + 1$ ، $j = 1, 2, ..., \max t - 1$ و $T_{\max} = v$. در این تحقیق قاعده FIFO اتخاذ می گردد. زمانی که لیست ممنوع پر است، حرکت جدید جایگزین اولین ورودی لیست ممنوع می شود.

۵. نتایج محاسباتی

۵-۱. تست های آزمایشی

محیط آزمایشگاهی و مفهوم هر پارامتر به صورت زیر شرح داده می‌شود. n تعداد کارها، m تعداد ماشین ها و L تعداد سطح هاست. مسأله $n \times m \times L$ یک مسأله RPFS است با n کار، m ماشین و L سطح. مسایل آزمایشگاهی به سه دسته طبقه بندی می شوند: مسایل کوچک، مسایل متوسط، مسایل بزرگ.

انواع مسایل کوچک شامل ۳×۳×۳، ۴×۴×۴، ۳×۴×۵، ۴×۵×۵، ۵×۸×۶، ۴×۸×۷، ۴×۸×۸، ۴×۷×۹، ۳×۹×۹ و ۳×۶×۱۰ می شوند. مسایل متوسط شامل ۵×۱۷×۱۱، ۶×۲۰×۱۲، ۷×۱۹×۱۳، ۹×۱۸×۱۴، ۶×۱۷×۱۵، ۷×۱۶×۱۶، ۸×۱۵×۱۷، ۶×۱۶×۱۸، ۱۰×۱۲×۱۹ و ۸×۱۵×۲۰ می‌شوند. مسایل بزرگ شامل ۱۰×۲۵×۲۵، ۷×۳۰×۳۰، ۶×۴۰×۴۰، ۵×۵۰×۵۰ و ۳×۶۰×۶۰ می شوند. زمان پردازش هر عملیات برای هر نوع از این مسایل یک عدد تصادفی تولید شده در بازه ی [۱,۱۰۰] است. از آنجایی که زمان-های پردازش بیشتر مسایل بنچ مارک در این دامنه ایجاد می شوند [۲۹]. مدل برنامه ریزی عدد صحیح با نرم افزار LINGO 8.0 حل شده و همه الگوریتم ها در زبان برنامه نویسی MATLAB 7.0 تحت سیستم عامل مایکروسافت ویندوز ایکس پی و با CPU دوهسته ای ۱/۸۰ گیگاهرتز و رم ۰/۹۹ گیگا بایت کد و اجرا شده است.

۵-۲. محاسبه موعد تحویل

موعدهای تحویل به طور یکنواخت در بازه $\left[P\left(1-T-R/2\right); P\left(1-T+R/2\right)\right]$ توزیع می شود. این روش توزیع موعدهای تحویل در ادبیات بسیار رایج است. T فاکتور دیرکرد و R حدود موعد تحویل نامیده می شود. P یک حد پایین برای زمان تکمیل کارهاست [۳۰] که به صورت معادله (۲۰) برای مسأله مورد بررسی در این مقاله، بدست می آید:

$$\max\left\{\begin{array}{l}\max\limits_{1\le j\le m}\left\{\sum\limits_{k=1}^{nL} t_{kj} + \min\limits_{k}\sum\limits_{v=1}^{j-1} t_{kv} + \min\limits_{k}\sum\limits_{v=j+1}^{m} t_{kv}\right\},\\ \max\limits_{1\le k\le nL}\sum\limits_{j=1}^{m} t_{kj}\end{array}\right\} \qquad (۲۰)$$

ترتیب پردازش هر کار روی ماشین ها L بار تکرار می شود. بنابراین هر کار دارای L زیر-کار است و در کل nL زیر-کار داریم. در فرمول بالا، k شمارنده ی زیر-کارهاست. t_{kj} زمان پردازش زیر-کار k روی ماشین M_j است. با این تعریف می توان مسأله جریان کارگاهی دوباره وارد شونده را با مسأله جریان کارگاهی غیر دوباره وارد شونده مطابقت داده و حد پایینی برای زمان تکمیل کارها بدست آورد. چهار سناریو برای موعدهای تحویل با ترکیبات مختلف برای T و R به صورت زیر می باشد:

- سناریو ۱: T=۰/۲ و R=۰/۶،
- سناریو ۲: T=۰/۲ و R=۱/۲،
- سناریو ۳: T=۰/۴ و R=۰/۶،
- سناریو ۴: T=۰/۴ و R=۱/۲،

برای هر ترکیب از تعداد کارها (n)، تعداد ماشین ها (m) تعداد سطح ها (L) و ماتریس زمان های پردازش، ما ۴ مثال طبق سناریوهای تعریف شده در بالا ایجاد می کنیم. نتایج ارایه شده حاصل میانگین برای این ۴ مثال می باشد.

۳-۵. ارزیابی نتایج محاسباتی

GA، TS و SA را تحت معیار کمینه سازی $T_{\max}$ با یکدیگر مقایسه می کنیم. چون تابع هدف از نوعی است که ممکن است مقدار جواب بهینه به صفر برسد، برای مقایسه اندازه گیری کارایی، شاخص انحراف نسبی (RDI) انتخاب می‌شود. RDI به صورت معادله (۲۱) محاسبه می گردد:

$$RDI = \frac{A\lg_{sol} - Min_{sol}}{Max_{sol} - Min_{sol}} \qquad (۲۱)$$

که $A\lg_{sol}$ مقدار هدف بدست آمده توسط الگوریتم در دست برای نمونه مورد نظر است. همچنین Min_{sol} و Max_{sol} به ترتیب بهترین و بدترین حل بدست آمده توسط هر کدام از الگوریتم هاست. واضح است که مقادیر کمتر RDI ارجح ترند. در مواردی که Max_{sol} و Min_{sol} با یکدیگر برابرند (یعنی همه ی الگوریتم ها حل یکسانی را ایجاد می کنند)، RDI برای همه ی الگوریتم ها صفر خواهد شد.

۴-۵. نتایج محاسباتی

در این آزمایش ها برای از بین رفتن تبعات حاصل از تصادفی بودن نتایج، ۴ بار هر نوع مسأله اجرا می شود و $T_{\max}$ متوسط تحلیل می-گردد. مقدار کارامد یک پارامتر طبق پروسه زیر انتخاب می گردد. فرض کنید که در الگوریتم ژنتیک به دنبال مقدار کارامد برای پارامتر خاصی هستیم، پس چندین تست با ثابت نگه داشتن مقادیر پارامترهای دیگر و متغیر بودن آن پارامتر خاص انجام می دهیم. مقدار پارامتر مورد نظر با توجه به معیار کارایی الگوریتم انتخاب می‌شود. چنین فرایندی در هر الگوریتمی و برای بقیه پارامترها نیز به کار گرفته می شود. در تنظیم پارامترها به همگرایی نمودار حاصل شده از میانگین حل ها و بهترین حل ها نیز توجه می گردد. در الگوریتم ژنتیک، اندازه جمعیت ۱۰۰، احتمال تقاطع ۰/۸ و احتمال جهش ۰/۰۵ را بکار می گیریم. برای الگوریتم شبیه سازی تبرید مقدار N (تعداد مراحل سرد شدن) را برابر ۵۰ قرار می دهیم. و سرانجام اندازه لیست ممنوع در الگوریتم جستجوی ممنوع، ۷ در نظر گرفته می شود.

در سال های اخیر برخی پژوهشگران برای منصفانه شدن مقایسه کارایی الگوریتم ها، شرط توقف آنها را زمان های برابر در نظر گرفته اند. در این مطالعه نیز زمان های برابر به عنوان شرط توقف الگوریتم های مورد مقایسه در نظر گرفته شده است. با انجام اجراهای مختلف برای اندازههای مختلف مسأله با درنظر گرفتن تعداد مشخصی از تکرارهای متوالی که در آن، جواب بدست آمده در آخر تکرار الگوریتم ثابت باقی بماند، زمان اجرا به طور تجربی حاصل می شود. زمان ها بر حسب ثانیه می باشند.

۱-۴-۵. مسایل کوچک

ابتدا برای هر نوع مسأله، متوسط $T_{\max}$ الگوریتم های فراابتکاری با زمان تکمیل برنامه بهینه مقایسه می شود که نتایج در جدول ۲ نشان داده شده است. این نتایج نشان می دهد که GA کاراتر است و کیفیت حل خوبی دارد. درصد خطای الگوریتم ها بدین صورت تعریف می شود:

$$\text{Percentage error} = \frac{T_{\max}(A\lg) - T_{\max}(IP)}{T_{\max}(IP)} \qquad (۲۲)$$

که $T_{\max}(A\lg)$ و $T_{\max}(IP)$ به ترتیب زمان تکمیل برنامه بدست آمده توسط الگوریتم فراابتکاری و IP هستند.

جدول ۲ . نتایج مقایسه مسایل کوچک حل شده با برنامه ریزی عدد صحیح و الگوریتم های فراابتکاری

اندازه مسأله	GA	SA	TS
۳×۳×۳	۰/۰۰	۰/۰۰	۰/۰۰
۴×۴×۴	۰/۰۰	۰/۰۰	۱/۱۸
۵×۴×۳	۰/۰۰	۰/۰۰	۲/۷۸
۵×۵×۴	۰/۰۰	۰/۳۱	۱/۹۳
۶×۸×۵	۰/۰۰	۰/۰۰	۱/۹۶
۷×۸×۴	۰/۰۴	۱/۰۶	۱/۹۴
۸×۸×۴	۰/۷۸	۱/۳۹	۱/۴۶
۹×۷×۴	۰/۷۹	۱/۰۶	۱/۳۶
۹×۹×۳	۰/۶۷	۰/۷۵	۱/۱۹
۱۰×۶×۳	۰/۹۰	۰/۹۶	۲/۲۴
میانگین	۰/۳	۰/۶	۱/۶

در این آزمایشات، RDI برای هر سه الگوریتم محاسبه شده و میانگین این مقادیر نیز برای ۱۰ مسأله نمونه آورده شده است. نتایج در جدول ۳ نشان داده شده است. الگوریتم ژنتیک با داشتن متوسط RDI کمتر، دارای کارایی بهتری نسبت به دو الگوریتم دیگر برای مسایل کوچک می باشد.

جدول ۳ . نتایج مقایسات الگوریتم ها برای مسایل کوچک

زمان اجرا	TS	SA	GA	اندازه مسأله
۰/۹	۰/۰۰۰۰	۰/۰۰۰۰	۰/۰۰۰۰	۳×۳×۳
۲/۰	۰/۰۷۵۰	۰/۰۰۰۰	۰/۰۰۰۰	۴×۴×۴
۲/۴	۰/۲۶۸۳	۰/۰۰۰۰	۰/۰۰۰۰	۵×۴×۳
۳/۶	۰/۷۵۷۶	۰/۱۲۱۲	۰/۰۰۰۰	۵×۵×۴
۱۳/۶	۰/۹۰۰۰	۰/۰۰۰۰	۰/۰۰۰۰	۶×۸×۵
۱۷/۹	۰/۷۷۱۷	۰/۳۴۳۷	۰/۰۰۹۸	۷×۸×۴
۲۴/۰	۰/۵۸۸۰	۰/۴۶۰۴	۰/۲۰۱۲	۸×۸×۴
۲۵/۶	۰/۶۱۴۹	۰/۳۶۷۸	۰/۱۸۸۱	۹×۷×۴
۲۳/۹	۰/۷۸۵۷	۰/۱۹۲۹	۰/۰۸۳۵	۹×۹×۳
۲۱/۸	۰/۴۰۷۴	۰/۱۴۳۴	۰/۱۰۳۷	۱۰×۶×۳
۱۳/۵	۰/۵۱۶۹	۰/۱۶۲۹	۰/۰۵۸۶	میانگین

۵-۴-۲. مسایل متوسط

برای مسایل متوسط برای ۱۰ اندازه مختلف، مسایل نمونه با هدف کمینه سازی $T_{\max}$ اجرا شده و نتایج در جدول ۴ ارایه شده است.

جدول ۴ . نتایج مقایسات الگوریتم ها برای مسایل متوسط

زمان اجرا	TS	SA	GA	اندازه مسأله
۲۰۸/۷	۰/۶۴۷۸	۰/۶۳۶۷	۰/۱۲۹۳	۱۱×۱۷×۵
۳۷۵/۲	۰/۷۹۱۹	۰/۶۷۷۰	۰/۱۷۵۸	۱۲×۲۰×۶
۴۱۰/۰	۰/۵۹۵۱	۰/۳۰۱۴	۰/۱۹۳۳	۱۳×۱۹×۷
۶۵۲/۹	۰/۵۵۹۹	۰/۰۹۴۶	۰/۱۱۱۸	۱۴×۱۸×۹
۴۰۹/۱	۰/۱۷۳۷	۰/۳۳۴۲	۰/۰۷۲۳	۱۵×۱۷×۶
۵۲۷/۷	۰/۶۹۳۳	۰/۴۱۷۴	۰/۰۸۲۲	۱۶×۱۶×۷
۶۵۷/۰	۰/۴۷۶۷	۰/۳۲۵۰	۰/۳۰۱۷	۱۷×۱۵×۸
۴۶۶/۹	۰/۵۳۶۶	۰/۳۷۴۴	۰/۲۹۴۵	۱۸×۱۶×۶
۷۶۴/۸	۰/۵۷۵۴	۰/۴۴۹۸	۰/۱۲۹۴	۱۹×۱۲×۱۰
۸۴۰/۸	۰/۶۱۸۸	۰/۶۳۰۰	۰/۶۰۱۳	۲۰×۱۵×۸
۵۳۱/۳	۰/۵۶۶۹	۰/۴۲۴۰	۰/۲۰۹۱	میانگین

همانطور که در جدول ۴ مشاهده می شود، الگوریتم ژنتیک با توجه به معیار اندازه گیری عملکرد مورد نظر برای مسایل متوسط، بهتر است.

۵-۴-۳. مسایل بزرگ

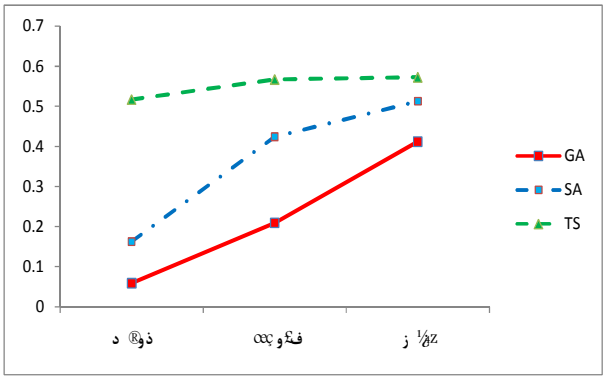
مسایل بزرگ با اساسی مشابه با دیگر مسایل، تست می شوند و انواع مسایل بزرگ تست شده در جدول ۵ نشان داده شده اند. برتری الگوریتم ژنتیک در آزمایشات مربوط به این مسایل نیز تأیید می-گردد.

جدول ۵. نتایج مقایسات الگوریتم ها برای مسایل بزرگ

زمان اجرا	TS	SA	GA	اندازه مسأله
۶۲۵۰	۰/۵۱۶۲	۰/۵۰۰۷	۰/۴۸۰۱	۲۵×۲۵×۱۰
۶۳۸۰	۰/۶۱۳۵	۰/۵۰۷۳	۰/۳۶۱۳	۳۰×۳۰×۷
۹۶۹۰	۰/۵۶۰۷	۰/۵۲۱۸	۰/۴۲۵۸	۴۰×۴۰×۶
۱۰۸۰۰	۰/۵۶۰۳	۰/۴۸۳۶	۰/۴۲۶۲	۵۰×۵۰×۵
۱۲۵۰۰	۰/۶۱۱۱	۰/۵۴۹۸	۰/۳۶۴۲	۶۰×۶۰×۳
۹۱۲۴	۰/۵۷۲۳	۰/۵۱۲۶	۰/۴۱۱۵	میانگین

۵-۴-۴. تحلیل نتایج با توجه به اندازه مسأله

در این قسمت، به منظور بررسی تأثیر تعداد کارها بر کارایی الگوریتم ها، مبنای مقایسات بر اساس اندازه مسأله قرار گرفته و نتایج حاصل از این مقایسه در شکل ۴ نشان داده شده است. چنانکه در این شکل مشاهده می شود، در تمام اندازها الگوریتم ژنتیک دارای کارایی بهتری نسبت به سایر الگوریتم ها می باشد. الگوریتم شبیه سازی تبرید نیز نسبت به الگوریتم جستجوی ممنوع بهتر می باشد.



شکل ۴. نمودار متوسط درصد انحراف نسبی برای الگوریتم‌های مختلف

۶. نتیجه گیری و پیشنهادها

با توجه به اهمیت معیار حداکثر دیرکرد (T_{max}) در ادبیات زمانبندی و نبود بررسی روش‌های فراابتکاری برای حل مسأله جریان کارگاهی جایگشتی دوباره وارد شونده (RPFS) جند ماشینه با این معیار در ادبیات، این تحقیق مدل ریاضی مسأله را توسعه داده و سه الگوریتم فراابتکاری را برای حل آن به کار می‌گیرد. در RPFS کد گذاری بر پایه کار برای بررسی انواع مختلف مسایل انجام می‌شود. نتایج نشان می‌دهد که الگوریتم ژنتیک در مقایسه با الگوریتم‌های شبیه‌سازی تبرید و جستجوی ممنوع، حل‌های مطلوبی را در یک زمان قابل قبول برای مسایل با اندازه‌های مختلف بدست می‌دهد. برای مسایل کوچک، درصد پیدا کردن حل های بهینه GA بیشتر است.

توسعه و بهبود الگوریتم‌های فراابتکاری با ترکیب الگوریتم‌های ابتکاری و تحقیق کامل برای تعیین پارامترهای مناسب، برای تحقیقات آتی پیشنهاد می‌شود. همچنین می‌توان از روش‌های طراحی آزمایش برای پی بردن به بهترین عملگرها و روش جستجوی همسایه استفاده نمود. ایجاد فرض های جدید همانند در نظر گرفتن احتمال خرابی ماشین‌ها، اضافه کردن زمان‌های آماده-سازی، در نظر گرفتن زمان حمل و نقل مواد بین ایستگاه‌های کاری و یا هر فرض جدیدی که باعث واقعی‌تر شدن مسأله شود، به عنوان زمینه‌های جدید برای تحقیقات آینده پیشنهاد می‌شود. بکارگیری معیارهای کارایی دیگر و مقایسه با سایر الگوریتم‌های بهینه‌سازی نیز می‌تواند زمینه‌ای برای بررسی بیشتر باشد.

مراجع

[1] Baker, K.R., *Introduction to Sequencing and Scheduling*. New York: John Wiley & Sons, 1974.

[2] Wang, M.Y., Sethi, S.P., Van de Velde, S.L., "*Minimizing Makespan in a Class of Re-Entrant Shops",* Operations Research, Vol.45, No.5, 1997, pp.702–712.

[3] Pinedo, M., *Scheduling: Theory, Algorithms, and Systems.* Prentice-Hall, New Jersey, 2002.

[4] Uzsoy, R., Lee, C.Y., Martin-Vega, L.A., "*A Review of Production Planning and Scheduling Models in the Semiconductor Industry, Part 1: System Characteristics, Performance Evaluation and Production Planning",* IIE Trans, Vol. 24, 1992, pp.47–60.

[5] Johnson, S.M., "*Optimal Two- and Three-Stage Production Schedules with Set up Times included"*, Nav Res Logistics Q, Vol. 1, 1954, pp. 61–68.

[6] Graves, S.C., Meal, H.C., Stefek, D., Zeghmi, A.H., *Scheduling of Reentrant Flow Shops*, Technical report, Sloan School of management, 1983.

[7] Gupta, A.K., Sivakumar, A.I., "*Job Shop Scheduling Techniques in Semiconductor Manufacturing",* International Journal of Advanced Manufacturing Technology, Vol. 27, No.11-12, 2006, pp.1163-1169.

[8] Pan, J.C.H., Chen, J.S., "*Mixed Binary Integer Programming Formulations for the Reentrant Job Shop Scheduling Problem",* Computers & Operations Research, Vol. 32, 2005, pp. 1197–1212.

[9] Pan, J.C., Chen, J.S., "*Minimizing Makespan in Re-Entrant Permutation Flow-Shops*", Journal of the Operational Research Society, Vol. 54, 2003, pp. 642–53.

[10] Chen, J.S., "*A Branch and Bound Procedure for the Reentrant Permutation Flow-Shop Scheduling problem",* International Journal of Advanced Manufacturing Technology, Vol. 29, 2006, pp. 1186–1193.

[11] Chen, J.S., Pan, J.C.H., Lin, C.M., "*A Hybrid Genetic Algorithm for the Re-Entrant Flow-Shop Scheduling Problem",* Expert Systems with Applications, Vol. 34, 2008, pp. 570-577.

[12] Chen, J.S., Pan, J.C.H., Wu, C.K., **"***Minimizing Makespan in Reentrant Flow-Shops using Hybrid Tabu Search",* International Journal of Advanced Manufacturing Technology, Vol. 34, 2007, pp. 353–361.

[13] Chen, J.S., Pan, J.C.H., Wu, C.K., "*Hybrid Tabu Search for Re-Entrant Permutation Flow-Shop Scheduling Problem"*, Expert Systems with Applications, Vol. 34, 2008, pp. 1924–1930.

[14] Liu, M., Wu, C., "*Genetic Algorithm using Sequence Rule Chain for Multi-Objective Optimization in re-Entrant micro-Electronic Production Line"*, Robotics and Computer-Integrated Manufacturing, Vol. 20, 2004, pp. 225–236.

[15] Drobouchevitch, I.G., Strusevich, V.A., "*A Heuristic Algorithm for Two-Machine Re-Entrant Shop Scheduling",* Annals of Operations Research, Vol. 86, 1999, pp. 417–439.

[16] Choi, S.W., Kim, Y.D., "*Minimizing Makespan on an m-Machine Re-Entrant Flowshop",* Computers and Operations Research, Vol. 35, 2008, pp. 1684–1696.

[17] Kang, Y.H., Kim, S.S., Shin, H.J., "*A Scheduling Algorithm for the Reentrant Shop: an Application in Semiconductor Manufacture",* International Journal of Advanced Manufacturing Technology, Vol. 35, No. 5, 2007, pp. 566-574.

[18] Mönch, L., Drieÿel, R., "*A Distributed Shifting Bottleneck Heuristic for Complex Job Shops",* Computers and Industrial Engineering, Vol. 49, No. 3, 2005, pp. 363-380.

[19] Kuroda, M., Tomita, T., Maeda, K., "*Dynamic Control of a Cellular-Line Production System Under Variations in the Product Mix",* International Journal of Production Economics, Vol. 60-61, 1999, pp. 439-445.

[20] Choi, S.W., Kim, Y.D., Lee, G.C., "*Minimizing Total Uardiness of Orders with Reentrant Lots in a Hybrid Flowshop",* International Journal of Production Research,Vol. 43, No. 11, 2005, pp. 2149–2167.

[21] Demirko, lE., Uzsoy, R., "*Decomposition Methods for Re-Entrant Flow Shops with Sequence-Dependent Setup Times",* Journal of Scheduling, Vol. 3, No. 3, 2000, pp. 155–177.

[22] Choi, S.W., Kim, Y.D., "*Minimizing Total Tardiness on a Two-Machine Re-Entrant Flowshop",* European Journal of Operational Research, Vol. 199, 2009, pp. 375-384.

[23] Koulamas, C., "*The Total Tardiness Problem: Review and Extensions",* Operations Research, Vol. 42, 1994, pp. 1025–1041.

[24] Holland, J., *Adaptation in Natural and Artificial Systems*, Ann Arbor: University of Michigan Press, 1975.

[25] Gillies, A., *Machine Learning Procedures for Generating Image Domain Feature Detectors*, Ann Arbor: University of Michigan Press, 1985.

[26] Murata, T., Ishibuchi, H., Tanaka, H., *Genetic Algorithms for Flows Shop Scheduling Problems,* Computers & Industrial Engineering, Vol. 30, No. 4, 1996, pp.1061–1071.

[27] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., "*Optimization by Simulated Annealing",* Science, Vol. 220, 1983, pp. 671–680.

[28] Glover, F., "*Tabu Search – Part I*.", ORSA Journal on Computing, Vol. 1, 1989, pp. 190–206.

[29] Beasly, J.E., "*OR-Library: Distribution Test Problems by Electronic Mail"*, Journal of the Operational Research Society, Vol. 41 No.11, 1990, pp. 1069–1072.

[30] Choi, S.W., Kim, Y.D., "*Minimizing Makespan on an m-Machine Re-Entrant Flowshop",* Computers & Operations Research, Vol. 35, 2008, pp. 1684 – 1696.