

# ارائه یک الگوریتم رقابت استعماری کارآمد برای حل مسئله زمان بندی پروژه با محدودیت منابع

ایمان پناهی<sup>۱</sup>، نسیم نهاوندی<sup>۲\*</sup>

۱. دانش‌آموخته کارشناسی ارشد دانشکده مهندسی صنایع و سیستم‌ها، دانشگاه تربیت مدرس

۲. دانشیار دانشکده مهندسی صنایع و سیستم‌ها، دانشگاه تربیت مدرس

(تاریخ دریافت: ۹۵/۰۴/۲۳، تاریخ دریافت روایت اصلاح‌شده: ۹۵/۱۰/۰۴، تاریخ تصویب: ۹۶/۰۲/۰۴)

## چکیده

در این مقاله، الگوریتم جدیدی براساس چارچوب الگوریتم رقابت استعماری برای حل مسئله زمان بندی پروژه با محدودیت منابع ارائه می‌شود. در این مسئله، فعالیت‌های پروژه با توجه به محدودیت‌های منابع و روابط پیش‌نیازی، به گونه‌ای زمان بندی می‌شوند که زمان پروژه حداقل شود. در الگوریتم پیشنهادی، به منظور مدل‌سازی عملگر جذب، از عملگر تقاطع یکنواخت استفاده شده و برای جلوگیری از همگرایی ناقص الگوریتم، دو عملگر انقلاب یک نقطه‌ای و چند نقطه‌ای پیشنهاد شده است. همچنین به منظور جست‌وجوی بهتر فضای جواب، دو الگوریتم بهبود پیشرو-پس‌رو و الگوریتم جست‌وجوی محلی مبتنی بر جایگشت به کار رفته است. پارامترهای الگوریتم، به وسیله طراحی آزمایش‌های تنظیم و کارایی الگوریتم با حل مجموعه مسائل PSPLIB ارزیابی شده است. نتایج محاسبات و مقایسه آن‌ها با الگوریتم‌های موجود نشان می‌دهد که الگوریتم پیشنهادی، قابلیت یافتن جواب‌های نزدیک به بهینه در مسائل کوچک و تولید جواب‌های رقابتی در مسائل بزرگ را دارد.

**واژه‌های کلیدی:** الگوریتم بهینه‌سازی، الگوریتم رقابت استعماری، مسئله زمان بندی پروژه با محدودیت منابع.

## مقدمه

در مسئله زمان بندی پروژه با محدودیت منابع<sup>۱</sup> (RCPSP)، زمان شروع فعالیت‌های پروژه، با توجه به محدودیت منابع و روابط پیش‌نیازی تعیین می‌شود [۱]. این مسئله کاربردهای فراوانی در صنایع و رشته‌های مختلف مهندسی مانند تحقیقات پزشکی [۲]، توسعه نرم‌افزار [۳] و برنامه‌ریزی حسابرسی [۴] دارد. از سوی دیگر، بسیاری از مسائل زمان بندی مانند زمان بندی کار کارگاهی<sup>۲</sup> و جریان کارگاهی<sup>۳</sup> را می‌توان نوع خاصی از مسئله RCPSP در نظر گرفت [۵]. بلازویچ و همکاران اثبات کردند که مسئله RCPSP را باید NP-hard دانست [۱]. از این رو، امکان حل دقیق این مسئله، برای اندازه‌های بزرگ وجود ندارد. بدین ترتیب، برای حل این مسئله، روش‌های ابتکاری و فراابتکاری ایجاد شده است.

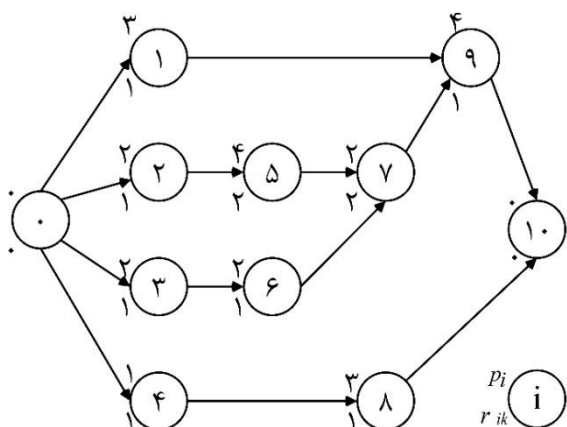
روش‌های ابتکاری، با به کارگیری قوانین مختلف اولویت باید بهترین جواب ممکن را برای مسئله RCPSP بیابند. تاکنون قوانین گوناگونی در اولویت مانند دیرترین زمان

شروع<sup>۴</sup> (LST)، دیرترین زمان پایان<sup>۵</sup> (LFT)، بیشترین تعداد پس‌نیاز<sup>۶</sup> (MTS) و... پیشنهاد شده است. دمولیمیستر و هرولن، هفتاد و چهار قانون اولویت را در پنج گروه مبتنی بر فعالیت، مبتنی بر شبکه، مبتنی بر مسیر بحرانی، مبتنی بر منابع و ترکیبی دسته بندی کردند [۶].

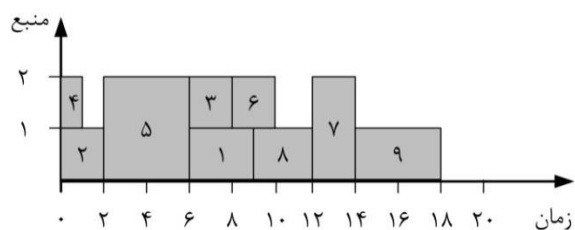
در دهه‌های اخیر، به دلیل کارایی بهتر الگوریتم‌های فراابتکاری در مقایسه با الگوریتم‌های ابتکاری، توجه بیشتری به آن‌ها شده است. هارتمان، الگوریتم ژنتیکی مبتنی بر جایگشت را طراحی کرد که در آن، جواب‌های اولیه، با قانون اولویت دیرترین زمان ساخته می‌شوند. وی همچنین برای ساخت جواب‌های موجه جدید، سه عملگر تقاطع یک نقطه‌ای، چند نقطه‌ای و یکنواخت پیشنهاد کرد [۷]. هارتمان الگوریتم ژنتیک انطباقی را طراحی کرد که در آن، یک ژن اضافه تعیین کننده الگوریتم کدگشایی جواب است [۸]. بیولیمن و لک الگوریتم تبرید شبیه‌سازی شده را پیشنهاد کردند که در آن، جواب‌ها به صورت فهرست فعالیت کدگذاری می‌شوند و تولید جواب‌های جدید، به وسیله جابه‌جایی فعالیت‌ها در فهرست

نشان می‌دهند.  $p_j$  مدت فعالیت  $j$  است. پس از آغاز فعالیت، امکان قطع شدن آن وجود ندارد. به دلایل منطقی و فناورانه، بین فعالیت‌ها روابط پیش‌نیازی وجود دارد. تا زمانی که تمامی پیش‌نیازهای فعالیت  $j$  به پایان نرسد، نمی‌توان فعالیت  $j$  را آغاز کرد؛ بنابراین، روابط پیش‌نیازی، از نوع پایان به شروع است. هر فعالیت، به تعداد مشخصی منبع تجدیدپذیر احتیاج دارد.  $K$  نوع منبع تجدیدپذیر وجود دارد که با  $k = 1, \dots, K$  نشان داده می‌شوند. میزان منبع در دسترس از نوع  $k$  در هر دوره  $b_k$  نشان داده می‌شود. فعالیت  $j$  به  $q_{jk}$  واحد از منبع  $k$  در هر دوره نیاز دارد که مقدار آن ثابت و مشخص است. تابع هدف مسئله، حداقل‌سازی زمان پروژه است.

در شکل ۱، مثالی ساده از مسئله RCPSP با ۱۱ فعالیت و یک نوع منبع تجدیدپذیر با ظرفیت ۲ واحد در هر دوره زمانی نشان داده شده است. شکل ۲ یک برنامه زمان‌بندی موجه را برای این مسئله نشان می‌دهد.



شکل ۱. مثالی از مسئله RCPSP [۱۲]



شکل ۲. یک برنامه زمان‌بندی موجه [۱۲]

فعالیت صورت می‌گیرد [۹]. والز و همکاران، الگوریتم ترکیبی ژنتیک را با یک الگوریتم جست‌وجوی محلی طراحی کردند. همچنین آن‌ها عملگر تقاطع اوج را معرفی کردند که مبتنی بر چگونگی تخصیص منابع در برنامه‌های زمان‌بندی است [۱۰]. زیارتی و همکاران، عملکرد سه الگوریتم زنبورعسل، الگوریتم کولانی زنبورعسل و الگوریتم بهینه‌سازی ازدحام زنبورها را برای مسئله RCPSP بررسی کردند [۱۱]. فنگ و ونگ، الگوریتم پرش قورباغه را پیشنهاد کردند. سازوکار تولید جواب‌ها در این الگوریتم، براساس چگونگی تخصیص منابع است. آن‌ها همچنین روش جدیدی برای ارزیابی سریع‌تر جواب‌های جدید معرفی کردند [۱۲]. فهمی و همکاران، به بررسی کارایی الگوریتم بهینه‌سازی ازدحام ذرات با تنظیم دوگانه مضاعف<sup>۷</sup> پرداختند [۱۳]. ژنگ و ونگ، الگوریتم بهینه‌سازی چندعاملی را برای حل مسئله زمان‌بندی پروژه طراحی کردند. نتایج این الگوریتم نشان داد که در مقایسه با سایر الگوریتم‌ها، این الگوریتم کارایی بسیار مناسبی دارد [۱۴].

آتش‌پز گرگری و لوکاس، الگوریتم رقابت استعماری<sup>۸</sup> را طراحی کردند [۱۵]. الگوریتم رقابت استعماری، یک روش بهینه‌سازی است که در آن، مانند الگوریتم ژنتیک - که از تکامل زیستی الهام گرفته است - از مفاهیم موجود در تکامل سیاسی - اجتماعی برای توسعه الگوریتم استفاده شده است.

از الگوریتم ICA برای حل بسیاری از مسائل بهینه‌سازی مهندسی مانند مسائل زمان‌بندی، مسئله بالانس خط مونتاژ و مسئله جانمایی تسهیلات استفاده شده است [۱۶]. اما این الگوریتم، برای حل مسئله RCPSP به کار نرفته است؛ بنابراین، هدف این تحقیق، توسعه الگوریتم رقابت استعماری، برای حل مسئله زمان‌بندی پروژه با محدودیت منابع است.

### مسئله زمان‌بندی پروژه با محدودیت منابع

در مسئله زمان‌بندی پروژه با محدودیت منابع، مسئله دارای  $n$  فعالیت است. فعالیت‌های ۰ و  $n + 1$  مجازی هستند که به مسئله افزوده شده‌اند و به ترتیب شروع و پایان پروژه را

## الگوریتم پایه رقابت استعماری

در الگوریتم رقابت استعماری، ابتدا کشورها به تعداد  $N_{pop}$  و به صورت تصادفی تولید می‌شوند. پس از محاسبه، مقدار تابع هدف کشورهای تولیدشده به تعداد  $N_{imp}$  از کشورهای اولیه که دارای کمترین هزینه هستند به عنوان استعمارگر، و سایر کشورها به عنوان مستعمره انتخاب می‌شوند. برای ساخت امپراتوری‌ها، کشورهای مستعمره به تناسب قدرت استعمارگرها بین آن‌ها تقسیم می‌شوند.

پس از تخصیص مستعمره‌ها به استعمارگران و تشکیل امپراتوری‌های اولیه، مستعمره‌ها به سمت استعمارگر خود حرکت می‌کنند. این حرکت، شبیه‌سازی سیاست جذبی<sup>۹</sup> است که به وسیله استعمارگران پیاده می‌شود. براساس تاریخ امپراتوری‌ها، استعمارگران سعی می‌کنند که مستعمره‌ها را از جنبه‌های فرهنگی، سیاسی و اجتماعی به سوی خود جذب کنند؛ درحالی که برخی از مستعمره‌ها در برابر این سیاست مقاومت می‌کنند و دست به اصلاحاتی می‌زنند. در الگوریتم رقابت استعماری، این عملگر به عنوان انقلاب<sup>۱۰</sup> شناخته می‌شود. همچنین این عملگر سبب جلوگیری از همگرایی سریع الگوریتم به جواب‌های بهینه محلی می‌شود.

پس از اعمال عملگر جذب و انقلاب بر مستعمره‌ها، ممکن است برخی از آن‌ها به موقعیت بهتری نسبت به استعمارگر دست پیدا کنند؛ بنابراین، باید جای مستعمره و استعمارگر عوض شود. در گام بعد، رقابت میان امپراتوری‌ها اتفاق می‌افتد. در این رقابت، امپراتوری‌ها تلاش می‌کنند که مستعمره‌های سایر امپراتوری‌ها را تصاحب کنند. طی این فرایند، امپراتوری ضعیف، ضعیف‌تر می‌شود و امپراتوری‌های قوی، قدرت بیشتری می‌یابند. در الگوریتم رقابت استعماری، ضعیف‌ترین امپراتوری، ضعیف‌ترین کشور خود را از دست می‌دهد و سایر امپراتوری‌ها متناسب با قدرتشان برای تصاحب این کشور رقابت می‌کنند. به این فرایند، رقابت بین امپراتوری می‌گویند. قدرت امپراتوری‌ها، به وسیله رابطه ۱ محاسبه می‌شود. در این رابطه،  $\xi$  عددی بین ۰ و ۱ است.

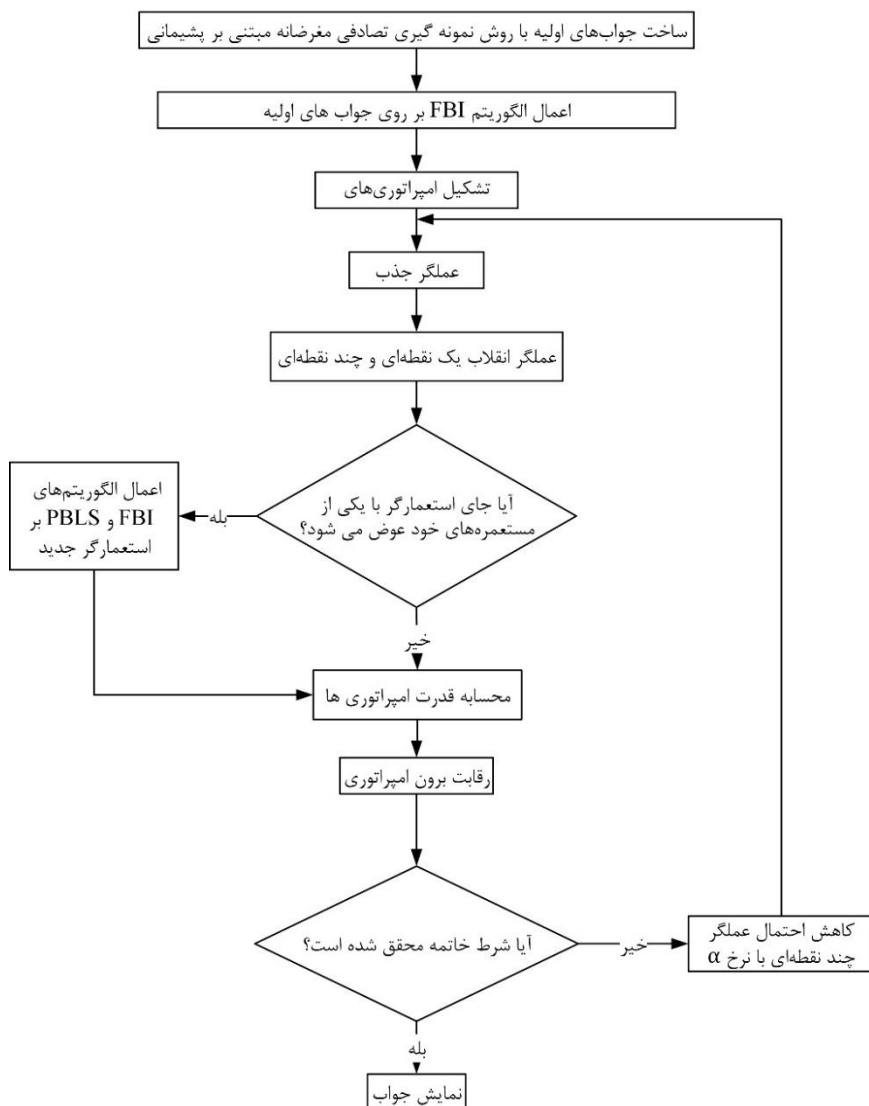
$$T.C_n = Cost(imerialist_n) + \xi \text{mean}\{Cost(colonies of empiri) \quad (1)$$

## الگوریتم رقابت استعماری پیشنهادی

رویه الگوریتم پیشنهادی ICA به این صورت است که ابتدا جواب‌های اولیه، به روش نمونه‌گیری تصادفی مغرضانه مبتنی بر پشیمانی<sup>۱۱</sup> تولید می‌شوند. سپس الگوریتم بهبود پیشرو-پس‌رو<sup>۱۲</sup> (FBI) روی جواب‌های تولیدشده اعمال می‌شود. در گام بعد، جمعیت اولیه براساس مقدار تابع تناسب زمان پروژه مرتب می‌شود و تعداد  $N_{imp}$  از بهترین جواب‌ها به عنوان استعمارگر انتخاب می‌شود. سایر جواب‌ها، به عنوان کلونی به هر استعمارگر تخصیص داده می‌شود. پس از تشکیل امپراتوری‌ها، هریک از کلونی‌ها به وسیله عملگر تقاطع یکنواخت، به سمت استعمارگر خود جذب می‌شود. در مرحله بعد، عملگر انقلاب یک نقطه‌ای و چندنقطه‌ای بر کلونی‌های هر امپراتوری اعمال می‌شود. اگر در هر امپراتوری، جواب بهتری نسبت به استعمارگر پیدا شد، جای آن جواب با استعمارگر عوض می‌شود و الگوریتم‌های FBI و جست‌وجوی محلی مبتنی بر جایگشت<sup>۱۳</sup> (PBLs) برای بهبود بیشتر استعمارگر جدید، بر آن اعمال می‌شوند. در نهایت، رقابت میان امپراتوری‌ها برای تصاحب بدترین جواب از بدترین امپراتوری شکل می‌گیرد. رویه این الگوریتم در شکل ۳ مشاهده می‌شود.

## کدگذاری و ساخت جواب‌های اولیه

در الگوریتم رقابت استعماری پیشنهادی، جواب‌ها به صورت فهرست فعالیت<sup>۱۴</sup> کدگذاری شده‌اند. فهرست فعالیت، توالی‌ای از فعالیت‌هاست که در آن، هر فعالیت قبل از پس‌نیازهایش درون فهرست قرار گرفته است. برای ایجاد جواب‌های اولیه، از روش نمونه‌گیری تصادفی مغرضانه مبتنی بر پشیمانی استفاده شده است. ساخت جواب‌ها ابتدا از یک فهرست خالی آغاز می‌شود. سپس در هر مرحله، از مجموعه فعالیت‌هایی که کاندید شده‌اند، یک فعالیت به صورت تصادفی انتخاب می‌شود و درون فهرست قرار می‌گیرد. فعالیت  $i$  کاندید است، اگر درون فهرست نیمه‌ساخته قرار نگرفته باشد و همه پیش‌نیازهای آن در فهرست نیمه‌ساخته وجود داشته باشد. برای ساخت هر فهرست، یکی از قوانین اولویت LFT، LTS و MTS، به صورت تصادفی انتخاب می‌شود. از قانون اولویت انتخاب‌شده، برای تعیین احتمال انتخاب فعالیت‌ها در هر مرحله استفاده می‌شود.



شکل ۳. چارچوب الگوریتم رقابت استعماری پیشنهادی

دلیل انتخاب سه قانون اولویت LFT، LST و MTS این است که به وسیله تولید جواب‌های اولیه متنوع، مناطقی از فضای جواب که احتمال می‌رود جواب بهینه در آنجا قرار داشته باشد، جست‌وجو شوند [۱۳]. علاوه بر این، کولیش نشان داده است که این سه قانون اولویت، در مقایسه با سایر قوانین عملکرد بهتری دارند. کولیش برای انتخاب فعالیت‌ها، روش نمونه‌گیری تصادفی مغرضانه مبتنی بر پشیمانی را با  $\alpha = \epsilon = 1$  پیشنهاد کرده است. از این رو، در این تحقیق نیز این دو پارامتر برابر با ۱ در نظر گرفته شده‌اند [۱۷]. پس از ساخت جواب‌های اولیه، الگوریتم FBI

احتمال انتخاب فعالیت  $j$  با اندازه اولویت  $\mu_j$  به وسیله رابطه ۲ محاسبه می‌شود.

$$\eta_j = \frac{(\mu_j + \epsilon)^\alpha}{\sum_{j \in D} (\mu_j + \epsilon)^\alpha} \quad (2)$$

اندازه اولویت فعالیت  $j$  برای قوانین اولویت LFT و LST با روابط ۳ و ۴، و برای قانون اولویت MTS با رابطه ۵ محاسبه می‌شود. در این روابط،  $D$  مجموعه فعالیت‌هایی است که کاندید شده‌اند.

$$\mu_j = \max_{i \in D} (LFT_i) - LFT_j + 1 \quad (3)$$

$$\mu_j = \max_{i \in D} (LST_i) - LST_j + 1 \quad (4)$$

$$\mu_j = MTS_j - \min_{i \in D} (MTS_i) + 1 \quad (5)$$

در نزدیک ترین زمان ممکن برای آغاز فعالیت زمان بندی می شوند؛ به طوری که پیش نیازهای فعالیت درون برنامه زمان بندی قرار گرفته باشند و منابع کافی برای اجرای فعالیت وجود داشته باشد.

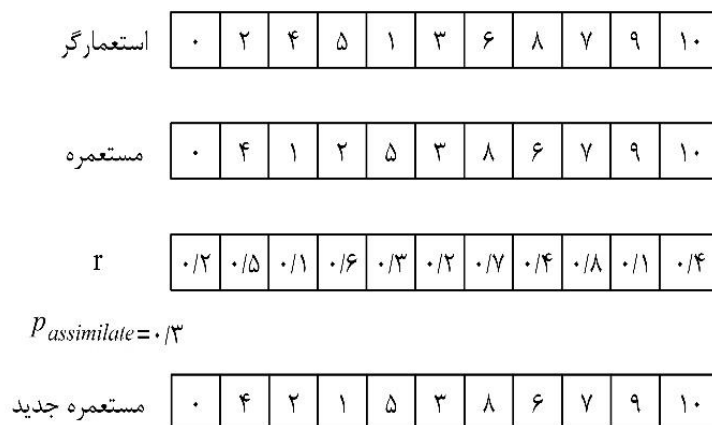
### عملگر جذب

به منظور مدل سازی سیاست جذب برای مسئله زمان بندی پروژه، از عملگر تقاطع یکنواخت پیشنهادی هارتمان [۷] استفاده شده است. این عملگر به این صورت است که ابتدا فهرستی جدید، به اندازه تعداد فعالیت های مسئله، از عددهای تصادفی  $r_i \in [0, 1]$  تولید می شود. سپس باید برای هر جایگاه از جواب جدید تصمیم گیری شود. اگر  $p_{assimilate} \geq r_i$  باشد، فعالیتی که دارای کمترین اندیس است و پیش تر در فهرست جدید قرار نگرفته است، از فهرست فعالیت استعمارگر انتخاب می شود و در جایگاه نام فهرست جدید قرار می گیرد. اگر  $p_{assimilate} < r_i$  باشد، فعالیتی که دارای کمترین اندیس است و پیش از این در فهرست جدید قرار نگرفته است، از فهرست فعالیت مستعمره انتخاب می شود و در جایگاه نام فهرست جدید قرار می گیرد. شکل ۴ چگونگی اجرای این عملگر را نشان می دهد.  $p_{assimilate}$  نرخ جذب و از پارامترهای الگوریتم است. جواب های جدید ایجاد شده به وسیله این عملگر همواره موجه هستند.

بر همه جواب ها اعمال می شود. در نهایت، بر اساس مقدار تابع هدف، استعمارگران مشخص می شوند و کشورها به عنوان مستعمره بین آن ها تقسیم می شوند.

### کدگشایی جواب ها

برای رمزگشایی جواب ها، دو الگوریتم در ادبیات زمان بندی پروژه پیشنهاد شده است. الگوریتم SSGS<sup>۱۵</sup> که بر اساس فعالیت ها و روابط پیش نیازی آن ها عمل می کند. روش دوم، الگوریتم رویه تولید برنامه زمان بندی موازی PSGS<sup>۱۶</sup> که نقاط تصمیم گیری آن مبتنی بر زمان است. تفاوت میان SSGS و PSGS در آن است که زمان بندی های تولید شده به وسیله SSGS، زمان بندی های فعال<sup>۱۷</sup> هستند. در حالی که زمان بندی های تولید شده به وسیله PSGS بدون تأخیر<sup>۱۸</sup> هستند. در زمان بندی های بدون تأخیر، زمان بیکاری فقط وقتی در برنامه زمان بندی ایجاد می شود که فعالیت موجه برای زمان بندی وجود نداشته باشد. زمان بندی بهینه، همواره بدون تأخیر نیست. از این رو، به کارگیری PSGS ممکن است به نادیده گرفتن جواب بهینه منجر شود؛ بنابراین، زمان بندی های بدون تأخیر، زیرمجموعه زمان بندی های فعال هستند. در مقابل، زمان بندی های فعال حداقل یک زمان بندی بهینه دارند [۱۷]؛ بنابراین، در این تحقیق، با توجه به این ویژگی مهم الگوریتم SSGS، از آن برای رمزگشایی جواب ها استفاده شده است. در الگوریتم SSGS، فعالیت ها به همان ترتیبی که در فهرست فعالیت ها قرار گرفته اند، وارد الگوریتم می شوند و



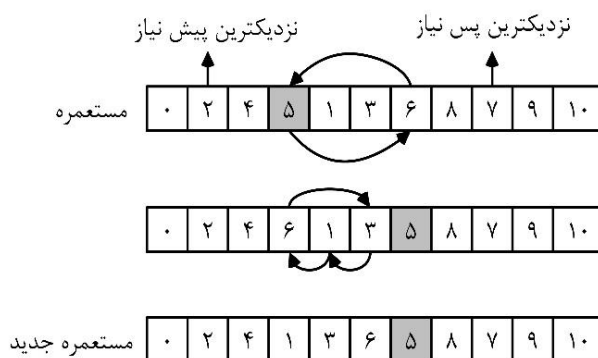
شکل ۴. عملگر جذب

شکل ۱ نشان می‌دهد. ابتدا انتخاب فعالیت ۵ به صورت تصادفی صورت می‌گیرد. نزدیک‌ترین پیش‌نیاز و پس‌نیاز فعالیت ۵ در این فهرست، به ترتیب فعالیت‌های ۲ و ۷ هستند؛ بنابراین، مکان جدید فعالیت ۵ باید از میان مکان فعالیت‌های ۴، ۱، ۳، ۶، ۸ به تصادف انتخاب شود که در اینجا فعالیت ۶ انتخاب شده است. پس از جابه‌جاشدن فعالیت ۵ با ۶، برای اینکه توالی فعالیت‌ها همچنان موجه بماند، باید جابه‌جایی چرخشی اعمال شود و فعالیت‌های ۱ و ۳ با همان ترتیب، پیش از فعالیت ۶ قرار گیرند. پس از اعمال این عملگر، جواب تولیدشده جدید همچنان موجه است.

## عملگر انقلاب تک‌نقطه‌ای

عملگر انقلاب تک‌نقطه‌ای به این صورت است که از فهرست فعالیت‌های هر کلونی، یک فعالیت به صورت تصادفی انتخاب می‌شود. این فعالیت را می‌توان در فاصله بین نزدیک‌ترین پیش‌نیاز و پس‌نیاز جابه‌جا کرد؛ بنابراین، مکان جدید فعالیت، به صورت تصادفی در این فاصله انتخاب می‌شود. سپس برای اینکه جواب جدید همچنان موجه بماند، یک جابه‌جایی چرخشی، روی فعالیت‌هایی که بین جایگاه‌های جدید و قدیم فعالیت وجود دارند، اعمال می‌شود.

شکل ۵ نحوه کارکرد این عملگر را برای جواب مسئله



شکل ۵. عملگر انقلاب یک‌نقطه‌ای

## رقابت میان امپراتوری‌ها

در رقابت میان امپراتوری‌ها، بدترین مستعمره از بدترین امپراتوری، انتخاب و به صورت تصادفی به سایر امپراتوری‌ها تخصیص داده می‌شود. در این پژوهش، برای انتخاب تصادفی امپراتوری، از روش انتخاب براساس چرخ رولت استفاده شده است. در صورتی که امپراتوری ضعیف، بدون کلونی و تنها دارای یک استعمارگر باشد، آن استعمارگر به صورت تصادفی به سایر امپراتوری‌ها تخصیص داده می‌شود. در نهایت، قدرت امپراتوری‌ها بار دیگر محاسبه می‌شود.

## روشن بهبود پیشرو - پس‌رو

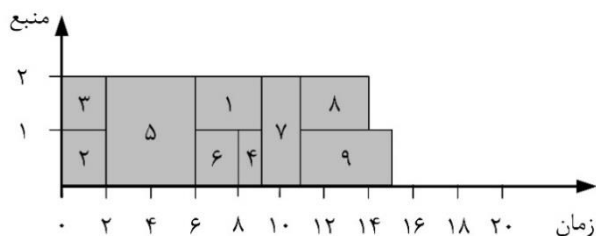
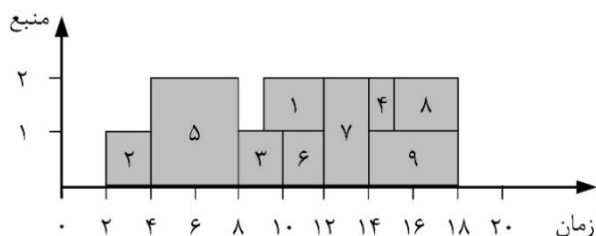
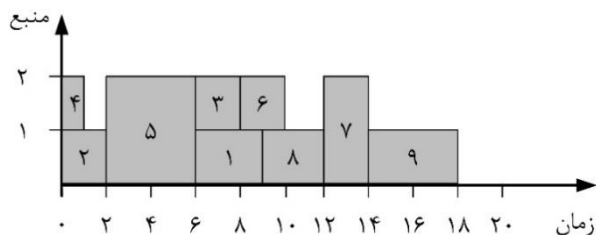
الگوریتم FBI روشی مؤثر برای بهبود برنامه زمان‌بندی

## عملگر انقلاب چندنقطه‌ای

این عملگر، فهرست فعالیت‌ها را به صورت کنترل‌شده به فهرست دیگری تبدیل می‌کند. عملگر چندنقطه‌ای به گونه‌ای طراحی شده است که در تکرارهای اولیه، انقلاب شدیدتری در مقایسه با تکرارهای نهایی ایجاد می‌کند. در هر فهرست فعالیت، اگر فعالیت  $i$  ام پیش‌نیاز فعالیت  $i + 1$  ام نباشد، فعالیت  $i$  ام با احتمال  $P_{revolution}$  جایگزین فعالیت  $i + 1$  ام می‌شود؛ بنابراین، ترتیب فعالیت‌ها تغییر می‌کند، اما جواب همچنان موجه است.  $P_{revolution}$  ابتدا برابر ۱ در نظر گرفته شده است و در هر تکرار،  $P_{revolution}$  با نرخ  $\alpha$  کم می‌شود.

می کند. به طور مشابه، زمان شروع فعالیتها که با محاسبات پسرو به دست آمده است، اولویت فعالیتها را برای محاسبات پیشرو بعدی تعیین می کند. شکل ۶ یک تکرار از الگوریتم FBI را برای بهبود زمان بندی شکل ۲ نشان می دهد.

است که به وسیله لی و ویلس معرفی شده است [۱۸]. به طور کلی در الگوریتم FBI، با به کارگیری متناوب SSGS به صورت پیشرو و پسرو، جوابها تا جایی بهبود پیدا می کنند که امکان بهبود بیشتر وجود نداشته باشد. زمان پایان فعالیتها که با محاسبات پیشرو به دست آمده است، اولویت فعالیتها را برای محاسبات پسرو بعدی تعیین



شکل ۶. یک تکرار از الگوریتم FBI [۱۲]

از مجموعه مسائل PSBLIB انتخاب شده اند. مسائل PSPLIB به وسیله نرم افزار ProGen تولید شده اند و این نرم افزار به دست کولیش و اسپریچر طراحی شده است [۱۹]. این مجموعه، ۴۸۰ مسئله با ۳۰ فعالیت و ۴۸۰ مسئله با ۶۰ فعالیت دارد.

برای مقایسه الگوریتم رقابت استعماری پیشنهادی با سایر الگوریتمها، از معیار متوسط درصد انحراف از حد پایین (Avg.LB.Dev) استفاده شده است که با رابطه ۶ محاسبه می شود.

$$(۶)$$

$$Avg.LB.Dev = \sum_{i=1}^R \left( \frac{Makespan_i - LB_i}{LB_i} \right) / R$$

## الگوریتم PBL

الگوریتم PBL مانند عملگر انقلاب چند نقطه ای است؛ با این تفاوت که در صورتی جای فعالیت  $i$  با فعالیت  $i + 1$  عوض می شود که فعالیت  $i$  پیش نیاز فعالیت  $i + 1$  نباشد و این جابه جایی، به بهبود زمان پروژه منجر شود. در این الگوریتم، همه جابه جایی های مجاز بررسی می شوند و در صورت بهبود جواب، این جابه جایی حفظ می شود.

## نتایج محاسباتی

به منظور ارزیابی عملکرد الگوریتم پیشنهادی، ۹۶۰ مسئله

مجموعه ۶۰ فعالیتی، به صورت تصادفی انتخاب شده‌اند. الگوریتم رقابت استعماری پیشنهادی، دارای پنج پارامتر تعداد کشورهای اولیه ( $N_{pop}$ )، تعداد امپراتوری‌ها ( $N_{imp}$ )، نرخ جذب ( $P_{assimilate}$ )، نرخ کاهش عملگر انقلاب چندنقطه‌ای ( $\alpha$ ) و ضریب ( $\xi$ ) است. این پارامترها و سطوح مختلف آن‌ها در جدول ۱ مشاهده می‌شود. براساس تعداد فاکتورها و تعداد سطح‌های آن‌ها، جدول طراحی آزمایش، با ۱۶ تیمار ( $L_{16}(4^5)$ ) تهیه شده است. طرح آزمایش‌ها در جدول ۴ قابل مشاهده است. Avg.LB.Dev متغیر پاسخ است و ارزیابی ۵۰۰۰ برنامه زمان‌بندی به عنوان شرط توقف الگوریتم در نظر گرفته شده است. با مشاهده روند تغییرات متغیر پاسخ در شکل‌های ۸ و ۹ می‌توان بهترین ترکیب از مقدار پارامترها را برای الگوریتم رقابت استعماری انتخاب کرد. این ترکیب در جدول ۲ مشاهده می‌شود.

$Makespan_i$  مقدار تابع هدف به دست آمده به وسیله الگوریتم رقابت استعماری برای مسئله  $i$  ام است.  $LB_i$  حد پایین مسئله  $i$  ام و  $R$  تعداد مسائل است. برای مسائل ۳۰ فعالیتی، جواب بهینه به عنوان حد پایین آن‌ها در نظر گرفته شده است، اما از آنجاکه جواب بهینه مسائل مجموعه ۶۰ فعالیتی مشخص نیست، از حد پایان روش مسیر بحرانی<sup>۲۰</sup> برای محاسبه Avg.LB.Dev استفاده شده است.

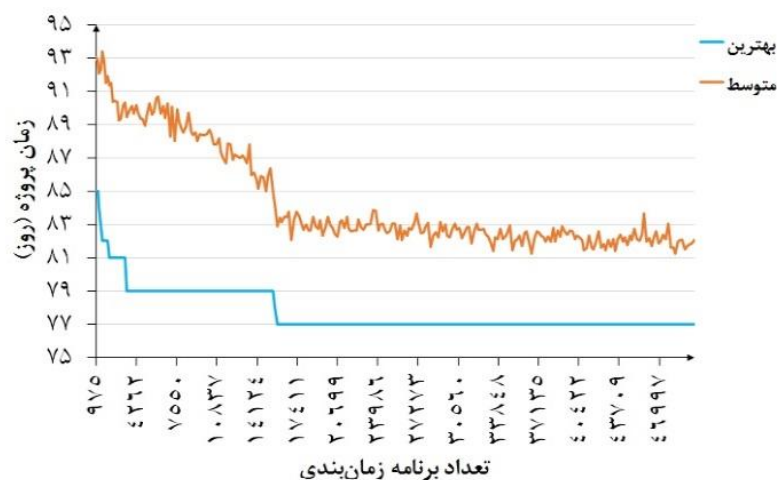
در ادامه، فرایند تنظیم پارامترهای ICA با طراحی آزمایش تاگوچی و نتایج حاصل از حل مسائل شرح داده می‌شود.

### تنظیم پارامترها

برای تنظیم پارامترهای الگوریتم رقابت استعماری، از طراحی آزمایش تاگوچی استفاده شده است. به همین جهت، ۴۸ مسئله از مجموعه ۳۰ فعالیتی و ۴۸ مسئله از

جدول ۱. ترکیب‌های مختلف از مقادیر فاکتورها

پارامتر	سطوح فاکتور			
	۱	۲	۳	۴
$N_{pop}$	۵۰	۱۰۰	۱۵۰	۲۰۰
$N_{imp}$	۵	۱۰	۱۵	۲۰
$P_{assimilate}$	۰/۲	۰/۴	۰/۶	۰/۸
$\alpha$	۰/۰۰۳	۰/۰۰۵	۰/۰۰۸	۰/۰۱
$\xi$	۰/۰۱	۰/۰۵	۰/۱	۱/۱۵



شکل ۷. نمودار همگرایی مسئله 605\_j



بهینه ۴۲۶، ۴۴۴ و ۴۷۴ مسئله از ۴۸۰ مسئله را بیابد. متوسط درصد انحراف از حد پایین برای مجموعه ۶۰ فعالیتی، به ازای معیار توقف ارزیابی ۱۰۰۰، ۵۰۰۰ و ۵۰۰۰۰ برنامه زمان بندی، به ترتیب ۱۲/۲۰، ۱۱/۷۴ درصد است. همچنین الگوریتم پیشنهادی ۳۱۵، ۳۵۷ و ۳۶۶ عدد از بهترین جواب های یافت شده برای ۴۸۰ مسئله را یافته است.

جدول ۳. نتایج الگوریتم ICA پیشنهادی

مجموعه مسائل	معیار توقف (تعداد برنامه های زمان بندی)		
	۱۰۰۰	۵۰۰۰	۵۰۰۰۰
۳۰ فعالیتی	۰/۳۰	۰/۱۳	۰/۰۱
۶۰ فعالیتی	۱۲/۵۳	۱۲/۲۰	۱۱/۷۴

### مقایسه عملکرد الگوریتم پیشنهادی با سایر الگوریتم ها

در این قسمت، الگوریتم ICA پیشنهادی با ۱۲ الگوریتم موجود در ادبیات مقایسه شده است. مقادیر به دست آمده به وسیله این الگوریتم ها، برای مجموعه مسائل ۳۰ و ۶۰ فعالیتی در جدول های ۵ و ۶ نشان داده شده است.

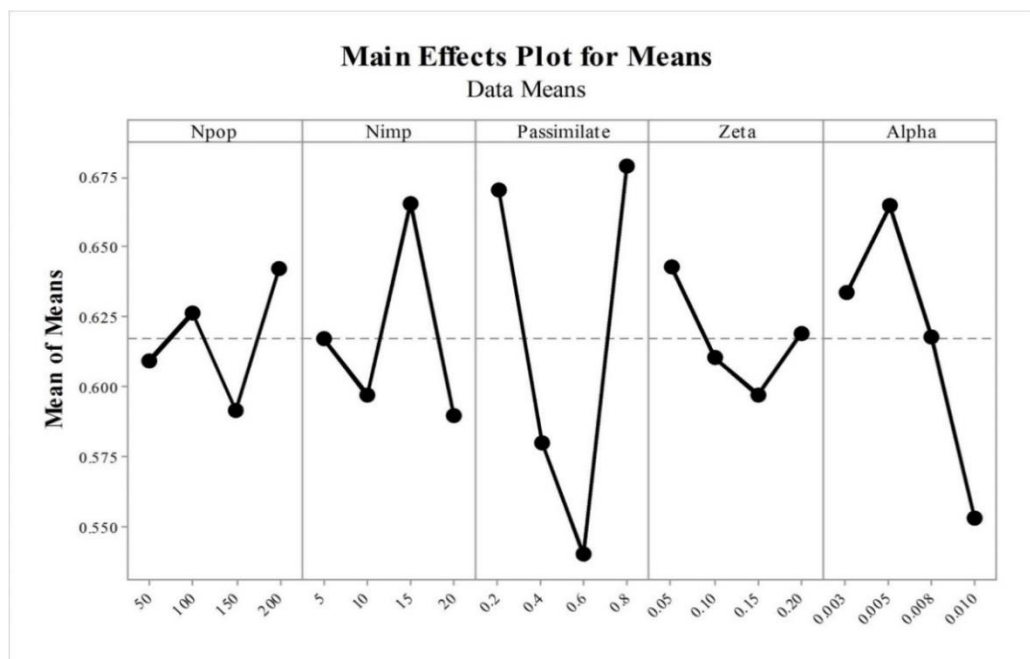
به منظور بررسی همگرایی الگوریتم ICA با پارامترهای انتخابی، یک مسئله به صورت تصادفی انتخاب، و نمودار همگرایی حل این مسائل رسم شده است (شکل ۷). این نمودار، بهترین زمان پروژه یافت شده به وسیله الگوریتم ICA و متوسط زمان پروژه همه کشورها را در هر تکرار از الگوریتم نشان می دهد. از آنجاکه الگوریتم نسبتاً به سرعت همگرا می شود، پارامترهای انتخاب شده برای الگوریتم مناسب اند.

جدول ۲. ترکیب انتخاب شده از پارامترها

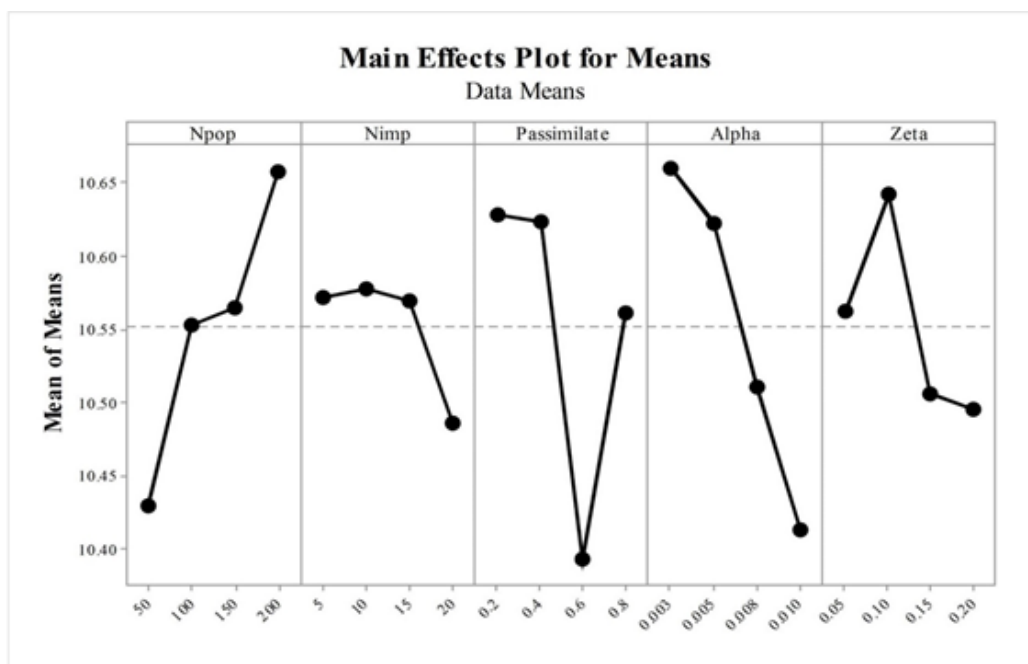
مجموعه مسائل	$N_{pop}$	$N_{imp}$	$P_{assimilate}$	$\alpha$	$\xi$
۳۰ فعالیتی	۱۵۰	۲۰	۰/۶	۰/۰۱	۰/۱۵
۶۰ فعالیتی	۵۰	۲۰	۰/۶	۰/۰۱	۰/۲

### نتایج محاسباتی الگوریتم پیشنهادی

جدول ۳ نتایج الگوریتم ICA پیشنهادی را نشان می دهد. متوسط درصد انحراف از جواب بهینه برای مجموعه ۳۰ فعالیتی، به ازای معیار توقف ارزیابی ۱۰۰۰، ۵۰۰۰ و ۵۰۰۰۰ برنامه زمان بندی به ترتیب برابر با ۰/۱۳، ۰/۰۱ و ۰/۰۱ درصد است. الگوریتم ICA قادر بوده است جواب



شکل ۸. روند تغییر پارامترها برای مسائل ۳۰ فعالیتی



شکل ۹. روند تغییر پارامترها برای مسائل ۶۰ فعالیتی

جدول ۴. طبقه‌بندی متعامد طراحی آزمایش‌ها و مقادیر پاسخ

Avg.LB.Dev		فاکتورها					شماره آزمایش
مجموعه ۶۰ فعالیتی	مجموعه ۳۰ فعالیتی	$\xi$	$\alpha$	$P_{assimilate}$	$N_{imp}$	$N_{pop}$	
۱۰/۶۴۶۷۳۷۸۶	۰/۷۰۴۰۵۸۹۹۳	۰/۰۵	۰/۰۰۳	۰/۲	۵	۵۰	۱
۱۰/۶۹۰۶۲۸۱۲	۰/۵۹۱۶۲۵۲۷۱	۰/۱	۰/۰۰۵	۰/۴	۱۰	۵۰	۲
۱۰/۲۰۱۲۶۵۱۴	۰/۵۶۰۳۰۵۶۳۱	۰/۱۵	۰/۰۰۸	۰/۶	۱۵	۵۰	۳
۱۰/۱۷۸۸۸۶۰۷	۰/۵۸۰۷۳۵۳۵	۰/۲	۰/۰۱	۰/۸	۲۰	۵۰	۴
۱۰/۵۴۷۸۹۸۵۲	۰/۵۹۰۸۸۰۰۴۹	۰/۲	۰/۰۰۸	۰/۴	۵	۱۰۰	۵
۱۰/۴۷۲۶۲۵۹۸	۰/۵۷۴۵۶۰۰۱۷	۰/۱۵	۰/۰۱	۰/۲	۱۰	۱۰۰	۶
۱۰/۷۸۲۰۱۵۷۸	۰/۷۴۶۲۲۴۴۱۷	۰/۱	۰/۰۰۳	۰/۸	۱۵	۱۰۰	۷
۱۰/۴۱۰۲۵۹۶۷	۰/۵۹۳۹۰۹۸۶۲	۰/۰۵	۰/۰۰۵	۰/۶	۲۰	۱۰۰	۸
۱۰/۳۷۸۱۳۳۴۱	۰/۴۴۱۴۷۰۴۲۴	۰/۱	۰/۰۱	۰/۶	۵	۱۵۰	۹
۱۰/۵۷۱۵۳۷۴۲	۰/۶۵۹۱۱۴۷۱۹	۰/۰۵	۰/۰۰۸	۰/۸	۱۰	۱۵۰	۱۰
۱۰/۶۷۴۸۴۷۴۵	۰/۷۴۲۹۸۳۲۶۴	۰/۲	۰/۰۰۵	۱/۲	۱۵	۱۵۰	۱۱
۱۰/۶۳۵۲۸۴۳۵	۰/۵۲۱۷۲۶۵۱۶	۰/۱۵	۰/۰۰۳	۱/۴	۲۰	۱۵۰	۱۲
۱۰/۷۱۴۶	۰/۷۳۱۳۰۵۲۹۵	۰/۱۵	۰/۰۰۵	۱/۸	۵	۲۰۰	۱۳
۱۰/۵۷۸۰۹۰۹۹	۰/۵۶۲۳۷۶۲۳۸	۰/۲	۰/۰۰۳	۱/۶	۱۰	۲۰۰	۱۴
۱۰/۶۲۰۱۸۱۴۳	۰/۶۱۴۴۵۰۷۳۹	۰/۰۵	۰/۰۱	۱/۴	۱۵	۲۰۰	۱۵
۱۰/۷۱۸۷۵۰۷۵	۰/۶۶۱۶۷۸۳۳۵	۰/۱	۰/۰۰۸	۱/۲	۲۰	۲۰۰	۱۶

جدول ۵. متوسط انحراف از جواب بهینه در مجموعه مسائل ۳۰ فعالیتی

الگوریتم	تعداد برنامه‌های زمان بندی		
	۵۰۰۰۰	۵۰۰۰	۱۰۰۰
الگوریتم پیشنهادی	۰/۰۱	۰/۱۳	۰/۳۰
نمونه برداری - انطباقی [۲۰]	-	۰/۵۲	۰/۷۴
ژنتیک [۷]	۰/۲۳	۰/۵۶	۱/۰۳
نمونه برداری - انطباقی [۲۱]	-	۰/۴۴	۰/۶۵
ژنتیک [۸]	-	۰/۲۲	۰/۳۸
تبرید شبیه سازی شده [۹]	-	۰/۲۳	۰/۳۸
ژنتیک [۲۲]	۰/۱۶	۰/۳۳	۰/۷۴
ژنتیک ترکیبی [۱۰]	۰/۰۲	۰/۰۶	۰/۲۷
زنبور عسل - FBI [۱۱]	۰/۰۴	۰/۱۹	۰/۴۲
نروژنتیک [۲۳]	-	۰/۱۰	۰/۱۳
جهش ترکیبی قورباغه [۱۲]	۰/۱۸	۰/۲۱	۰/۳۶
بهینه سازی ازدحام ذرات [۱۳]	۰/۰۲	۰/۰۵	۰/۲۲
بهینه سازی چند عاملی [۱۴]	۰/۰۱	۰/۰۶	۰/۱۷

رتبه عملکرد الگوریتم پیشنهادی در مقایسه با سایر الگوریتم‌ها، برای مجموعه ۳۰ فعالیتی و به ازای معیار توقف ارزیابی ۱۰۰۰، ۵۰۰۰ و ۵۰۰۰۰ برنامه زمان بندی به ترتیب ۴، ۵ و ۱ است.

جدول ۶. متوسط انحراف از جواب بهینه در مجموعه مسائل ۶۰ فعالیتی

الگوریتم	تعداد برنامه‌های زمان بندی		
	۵۰۰۰۰	۵۰۰۰	۱۰۰۰
الگوریتم پیشنهادی	۱۱/۷۴	۱۲/۲۰	۱۲/۵۳
نمونه برداری - انطباقی [۲۰]	-	۱۳/۰۶	۱۳/۵۱
ژنتیک [۷]	۱۲/۲۶	۱۲/۷۴	۱۳/۳۰
نمونه برداری - انطباقی [۲۱]	-	۱۲/۵۸	۱۲/۹۴
ژنتیک [۸]	-	۱۱/۷۰	۱۲/۲۱
تبرید شبیه سازی شده [۹]	-	۱۱/۹	۱۲/۷۵
ژنتیک [۲۲]	۱۲/۸۳	۱۳/۳۱	۱۳/۸
ژنتیک ترکیبی [۱۰]	۱۰/۷۳	۱۱/۱۰	۱۱/۵۶
زنبور عسل - FBI [۱۱]	۱۰/۸۴	۱۱/۱۹	۱۱/۸۹
نروژنتیک [۲۳]	-	۱۱/۲۹	۱۱/۵۱
جهش ترکیبی قورباغه [۱۲]	۱۰/۶۶	۱۰/۸۷	۱۱/۴۴
بهینه سازی ازدحام ذرات [۱۳]	۱۰/۸۵	۱۱/۱۹	۱۱/۸۶
بهینه سازی چند عاملی [۱۴]	۱۰/۶۴	۱۰/۸۴	۱۱/۶۴

SSGS استفاده شده است. جواب‌های اولیه با سه قانون اولویت LFT، LST و MTS و به‌وسیله روش نمونه‌گیری تصادفی مغرضانه مبتنی بر پیشیمانی ساخته شده‌اند. از عملگر تقاطع یکنواخت، به‌عنوان عملگر جذب استفاده شده است. همچنین دو عملگر انقلاب یک‌نقطه‌ای و چندنقطه‌ای پیشنهاد داده شده است. با به‌کارگیری دو الگوریتم جست‌وجوی محلی FBI و PBLs، جست‌وجوی فضای جواب بهبود یافته است. پارامترهای الگوریتم، به‌وسیله طراحی آزمایش تاگوچی تنظیم شده‌اند. در انتها، کارایی الگوریتم پیشنهادی، با حل مجموعه مسائل PSPLIB- که شامل ۴۸۰ مسئله ۳۰ فعالیتی و ۴۸۰ مسئله ۶۰ فعالیتی است- بررسی شده است. نتایج محاسباتی و مقایسه آن‌ها با ۱۲ الگوریتم موجود در ادبیات نشان می‌دهد که الگوریتم پیشنهادی، قابلیت یافتن جواب‌های نزدیک به بهینه برای مسائل کوچک را دارد و قادر است برای مسائل بزرگ‌تر جواب‌های رقابتی تولید کند؛ بنابراین، برای حل مسئله زمان‌بندی پروژه با محدودیت منابع، الگوریتمی کارا به‌شمار می‌رود.

در مجموعه ۶۰ فعالیتی، الگوریتم ICA با معیار توقف ارزیابی ۱۰۰۰ برنامه زمان‌بندی هشتمین الگوریتم، با معیار ۵۰۰۰ برنامه زمان‌بندی، نهمین الگوریتم و با معیار توقف ارزیابی ۵۰۰۰۰ برنامه زمان‌بندی، ششمین الگوریتم است. براساس تعریف کولیش و هارتمان، برای مقایسه الگوریتم‌های فراابتکاری، الگوریتم A، الگوریتم B را مغلوب می‌کند، اگر به ازای همه معیارهای توقف و همه مجموعه مسائل، الگوریتم A بدتر از الگوریتم B نباشد و حداقل به ازای یکی از مجموعه مسائل و یک معیار توقف، الگوریتم A بهتر از B باشد [۲۴]. براساس این معیار و جدول‌های ۵ و ۶، الگوریتم رقابت استعماری پیشنهادی، به‌وسیله الگوریتم بهینه‌سازی چندعاملی [۱۴] مغلوب می‌شود و بر چهار الگوریتم [۷، ۲۰، ۲۱، ۲۲] غلبه می‌کند.

### نتیجه‌گیری

نوآوری اصلی این مقاله، الگوریتم رقابت استعماری جدید برای حل مسئله زمان‌بندی پروژه با محدودیت منابع است. در این الگوریتم، جواب‌ها به‌صورت فهرست فعالیت کدگذاری شده‌اند و برای کدگشایی جواب‌ها، از الگوریتم

### مراجع

- Blazewicz, J., Lenstra, J. and Rinnoy Kan, A. H. (1983). "Scheduling subject to resource classification and complexity constraint.", *Discret. Appl. Math.*, Vol. 5, No. 1, PP. 11–24.
- Hartmann, S. (1997). *Scheduling medical research experiments: an application of project scheduling methods*, Technical Report, University Kiel, Germany.
- Alba, E. and Francisco Chicano, J. (2007). "Software project management with Gas", *Information Sciences*, Vol. 177, No. 11, PP. 2380–2401.
- Dodin, B., Elimam, A. A. and Rolland, E. (1998). "Tabu search in audit scheduling.", *European Journal of Operational Research*, Vol. 106, No. 2–3, PP. 373–392.
- Sprecher, A. (1994). "Special cases", In *Resource-constrained project scheduling: Exact methods for the multi-mode case*, 1<sup>th</sup> Ed, PP. 10–18, Springer, Berlin, Germany.
- Demeulemeester, E. L. and Herroelen, W. S. (2002). *The Resource-Constrained Project Scheduling Problem*, In *Project Scheduling: A Research Handbook*, 1<sup>th</sup> Ed, PP. 203–342, Springer, Berlin, Germany.
- Hartmann, S. (1998). "A competitive genetic algorithm for resource-constrained project scheduling", *Naval Research Logistics*, Vol. 45, No. 6, PP. 733–750.
- Hartmann, S. (2002). "A self-adapting genetic algorithm for project scheduling under resource constraints", *Naval Research Logistics*, Vol. 49, No. 5, PP. 433–448
- Bouleimen, K. and Lecocq, H. (2003). "A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version", *European Journal of Operational Research*, Vol. 149, No. 2, PP. 268–281.

10. Valls, V., Ballestín, F. and Quintanilla, S. (2008). "A hybrid genetic algorithm for the resource-constrained project scheduling problem", *European Journal of Operational Research*, Vol. 185, No. 2, PP. 495–508.
11. Ziarati, K., Akbari, R. and Zeighami, V. (2011). "On the performance of bee algorithms for resource-constrained project scheduling problem", *Applied Soft Computing*, Vol. 11, No. 4, PP. 3720–3733.
12. Fang, C. and Wang, L. (2012). "An effective shuffled frog-leaping algorithm for resource-constrained project scheduling problem", *Computers & Operations Research*, Vol. 39, No. 5, PP. 890–901.
13. Fahmy, A., Hassan, T. M. and Bassioni, H. (2014). "Improving RCPSP solutions quality with Stacking Justification—Application with particle swarm optimization", *Expert Systems with Applications*, Vol. 41, No. 13, PP. 5870–5881.
14. Zheng, X. and Wang, L. (2015). "A multi-agent optimization algorithm for resource constrained project scheduling problem", *Expert Systems with Applications*, Vol. 42, No. 15–16, PP. 6039–6049.
15. Atashpaz-Gargari, E. and Lucas, C. (2007). "Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition", *IEEE Congress on Evolutionary Computation*, PP. 4661–4667.
16. Hosseini, S. and Al Khaled, A. (2014). "A survey on the Imperialist Competitive Algorithm metaheuristic: Implementation in engineering domain and directions for future research", *Applied Soft Computing*, Vol. 24, PP. 1078–1094.
17. Kolisch, R. (1996). "Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation", *European Journal of Operational Research*, Vol. 90, No. 95, PP. 320–333.
18. Li, K. Y. and Willis, R. J. (1992). "An iterative scheduling technique for resource-constrained project scheduling", *European Journal of Operational Research*, Vol. 56, No. 3, PP. 370–379.
19. Kolisch, R. and Sprecher, A. (1997). "PSPLIB - A project scheduling problem library", *European Journal of Operational Research*, Vol. 96, No. 1, PP. 205–216.
20. Kolisch, R. and Drexel, A. (1996). "Adaptive search for solving hard project scheduling problems", *Naval Research Logistics*, Vol. 43, No. 1, PP. 23–40.
21. Schirmer, A. (2000). "Case-based reasoning and improved adaptive search for project scheduling. Naval Research Logistics", *Naval Research Logistics*, Vol. 47, No. 3, PP. 201–222.
22. Coelho, J. and Tavares, L. (2005). "Comparative analysis of metaheuristics for the resource constrained project scheduling problem", *European Journal of Operational Research*, Vol. 165, PP. 375–386.
23. Agarwal, A., Colak, S. and Erenguc, S. (2011). "A Neurogenetic approach for the resource-constrained project scheduling problem", *Computers & Operations Research*, Vol. 38, No. 1, PP. 44–50.
24. Kolisch, R. and Hartmann, S. (2006). "Experimental investigation of heuristics for resource-constrained project scheduling: An update", *European Journal of Operational Research*, Vol. 174, No. 1, PP. 23–37.

### واژه های انگلیسی به ترتیب استفاده در متن

1. Resource Constrained Project Scheduling Problem (RCPSP)
2. Job Shop Scheduling
3. Flow Shop Scheduling
4. Latest Start Time (LST)
5. Latest Finish Time (LFT)
6. Most Total Successor (MTS)
7. Double-Justification
8. Imperialist Competitive Algorithm (ICA)
9. Assimilation
10. Revolution

- 
11. Regret-Based Biased Random Sampling Method
  12. Forward-Backward Improvement (FBI)
  13. Permutation Based Local Search (PBLs)
  14. Activity list
  15. Serial Schedule Generation Scheme (SSGS)
  16. Parallel Schedule Generation Scheme (PSGS)
  17. Active Schedule
  18. Non-Delay Schedule
  19. Roulette Wheel Selection
  20. Critical Path Method