ارائهٔ یک الگوریتم رقابت استعماری کارآمد برای حل مسئلهٔ زمان‌بندی پروژه با محدودیت منابع

ایمان پناهی[۱]، نسیم نهاوندی[۲*]

۱. دانش‌آموختهٔ کارشناسی ارشد دانشکدهٔ مهندسی صنایع و سیستم‌ها، دانشگاه تربیت‌مدرس

۲. دانشیار دانشکدهٔ مهندسی صنایع و سیستم‌ها، دانشگاه تربیت‌مدرس

(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 مسئلهٔ زمان‌بندی پروژه با محدودیت منابع ارائه می‌شود. در این مسئله، فعالیت‌های پروژه با توجه به محدودیت‌های منابع و روابط پیش‌نیازی، به‌گونه‌ای زمان‌بندی می‌شوند که زمان پروژه حداقل شود. در الگوریتم پیشنهادی، به‌منظور مدل‌سازی عملگر جذب، از عملگر تقاطع یکنواخت استفاده شده و برای جلوگیری از همگرایی ناقص الگوریتم، دو عملگر انقلاب یک‌نقطه‌ای و چندنقطه‌ای پیشنهاد شده است. همچنین به‌منظور جست‌وجوی بهتر فضای جواب، دو الگوریتم بهبود پیشرو-پس‌رو و الگوریتم جست‌وجوی محلی مبتنی‌بر جایگشت به‌کار رفته است. پارامترهای الگوریتم، به‌وسیلهٔ طراحی آزمایش تاگوچی تنظیم و کارایی الگوریتم با حل مجموعه مسائل PSPLIB ارزیابی شده است. نتایج محاسبات و مقایسهٔ 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، مسئلهٔ زمان‌بندی پروژه با محدودیت منابع.

مقدمه

در مسئلهٔ زمان‌بندی پروژه با محدودیت منابع (RCPSP)[۱]، 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]، توسعهٔ نرم‌افزار [۳] و برنامه‌ریزی حسابرسی [۴] دارد. از سوی دیگر، بسیاری از مسائل زمان‌بندی مانند زمان‌بندی کار کارگاهی[۲] و جریان کارگاهی[۳] را می‌توان نوع خاصی از مسئلهٔ RCPSP درنظر گرفت [۵]. بلازویچ و همکاران اثبات کردند که مسئلهٔ RCPSP را باید NP-hard دانست [۱]. از این‌رو، امکان حل دقیق این مسئله، برای اندازه‌های بزرگ وجود ندارد. بدین‌ترتیب، برای حل این مسئله، روش‌های ابتکاری و فراابتکاری ایجاد شده است.

روش‌های ابتکاری، با به‌کارگیری قوانین مختلف اولویت باید بهترین جواب ممکن را برای مسئلهٔ RCPSP بیابند. تاکنون قوانین گوناگونی در اولویت مانند دیرترین زمان شروع[۴] (LST)، دیرترین زمان پایان[۵] (LFT)، بیشترین تعداد پس‌نیاز[۶] (MTS) و... پیشنهاد شده است. دمولیمیستر و هرولن، هفتادوچهار قانون اولویت را در پنج گروه مبتنی‌بر فعالیت، مبتنی‌بر شبکه، مبتنی‌بر مسیر بحرانی، مبتنی‌بر منابع و ترکیبی دسته‌بندی کردند [۶].

در دهه‌های اخیر، به‌دلیل کارایی بهتر الگوریتم‌های فراابتکاری در مقایسه با الگوریتم‌های ابتکاری، توجه بیشتری به آن‌ها شده است. هارتمان، الگوریتم ژنتیکی مبتنی‌بر جایگشت را طراحی کرد که در آن، جواب‌های اولیه، با قانون اولویت دیرترین زمان شروع ساخته می‌شوند. وی همچنین برای ساخت جواب‌های موجه جدید، سه عملگر تقاطع یک‌نقطه‌ای، چندنقطه‌ای و یکنواخت پیشنهاد کرد [۷]. هارتمان الگوریتم ژنتیک انطباقی را طراحی کرد که در آن، یک ژن اضافه تعیین‌کنندهٔ الگوریتم کدگشایی جواب است [۸]. بیولیمن و لکک الگوریتم تبرید شبیه‌سازی‌شده را پیشنهاد کردند که در آن، جواب‌ها به‌صورت فهرست فعالیت کدگذاری می‌شوند و تولید جواب‌های جدید، به‌وسیلهٔ جابه‌جایی فعالیت‌ها در فهرست

* نویسندهٔ مسئول: تلفن و فکس: ۰۲۱۸۲۸۸۴۱۴۱ Email: n_nahavandi@modares.ac.ir

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 مبتنی‌بر 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 مسئلهٔ RCSPS 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 دوگانهٔ مضاعف[۷] پرداختند [۱۳]. ژنگ و ونگ، الگوریتم بهینه‌سازی چندعاملی را برای حل مسئلهٔ زمان‌بندی پروژه طراحی کردند. نتایج این الگوریتم نشان داد که در مقایسه با سایر الگوریتم‌ها، این الگوریتم کارایی بسیار مناسبی دارد [۱۴].

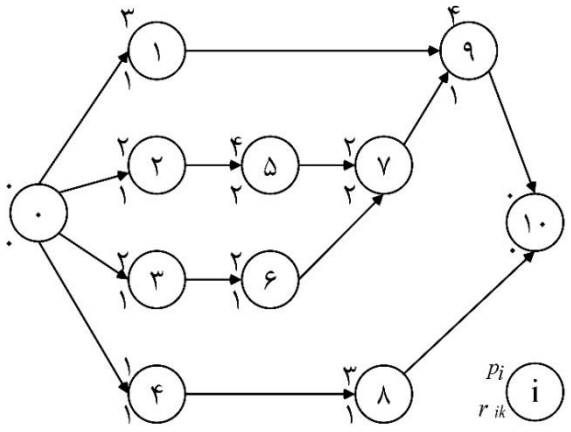
آتش‌پز گرگری و لوکاس، الگوریتم رقابت استعماری[۸] را طراحی کردند [۱۵]. الگوریتم رقابت استعماری، یک روش بهینه‌سازی است که در آن، مانند الگوریتم ژنتیک- که از تکامل زیستی الهام گرفته است- از مفاهیم موجود در تکامل سیاسی- اجتماعی برای توسعهٔ الگوریتم استفاده شده است.

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

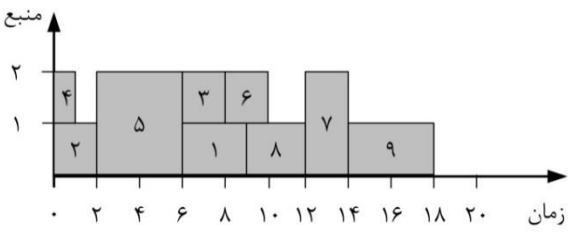
مسئلهٔ زمان‌بندی پروژه با محدودیت منابع

در مسئلهٔ زمان‌بندی پروژه با محدودیت منابع، مسئله دارای n فعالیت است. فعالیت‌های ۰ و $n+1$ مجازی هستند که به مسئله افزوده شده‌اند و به‌ترتیب شروع و پایان پروژه را نشان می‌دهند. p_j مدت فعالیت j است. پس از آغاز فعالیت، امکان قطع‌شدن 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 دردسترس از نوع k در هر دوره با 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 مسئلهٔ RCPSP با ۱۱ فعالیت و یک نوع منبع تجدیدپذیر با ظرفیت ۲ واحد در هر دورهٔ زمانی نشان داده شده است. شکل ۲ یک برنامهٔ زمان‌بندی موجه را برای این مسئله نشان می‌دهد.



شکل ۱. مثالی از مسئلهٔ RCPSP [۱۲]



شکل ۲. یک برنامهٔ زمان‌بندی موجه [۱۲]

الگوریتم پایهٔ رقابت استعماری

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

پس از تخصیص مستعمره‌ها به استعمارگران و تشکیل امپراتوری‌های اولیه، مستعمره‌ها به‌سمت استعمارگر خود حرکت می‌کنند. این حرکت، شبیه‌سازی سیاست جذبی[۹] است که به‌وسیلهٔ استعمارگران پیاده می‌شود. براساس تاریخ امپراتوری‌ها، استعمارگران سعی می‌کنند که مستعمره‌ها را از جنبه‌های فرهنگی، سیاسی و اجتماعی به‌سوی خود جذب کنند؛ درحالی‌که برخی از مستعمره‌ها دربرابر این سیاست مقاومت می‌کنند و دست به اصلاحاتی می‌زنند. در الگوریتم رقابت استعماری، این عملگر به‌عنوان انقلاب[۱۰] شناخته می‌شود. همچنین این عملگر سبب جلوگیری از همگرایی سریع الگوریتم به جواب‌های بهینهٔ 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 می‌یابد. 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 ببین امپراتوری می‌گویند. قدرت امپراتوری‌ها، به‌وسیلهٔ رابطهٔ ۱ محاسبه می‌شود. در این رابطه، ξ عددی بین ۰ و ۱ است.

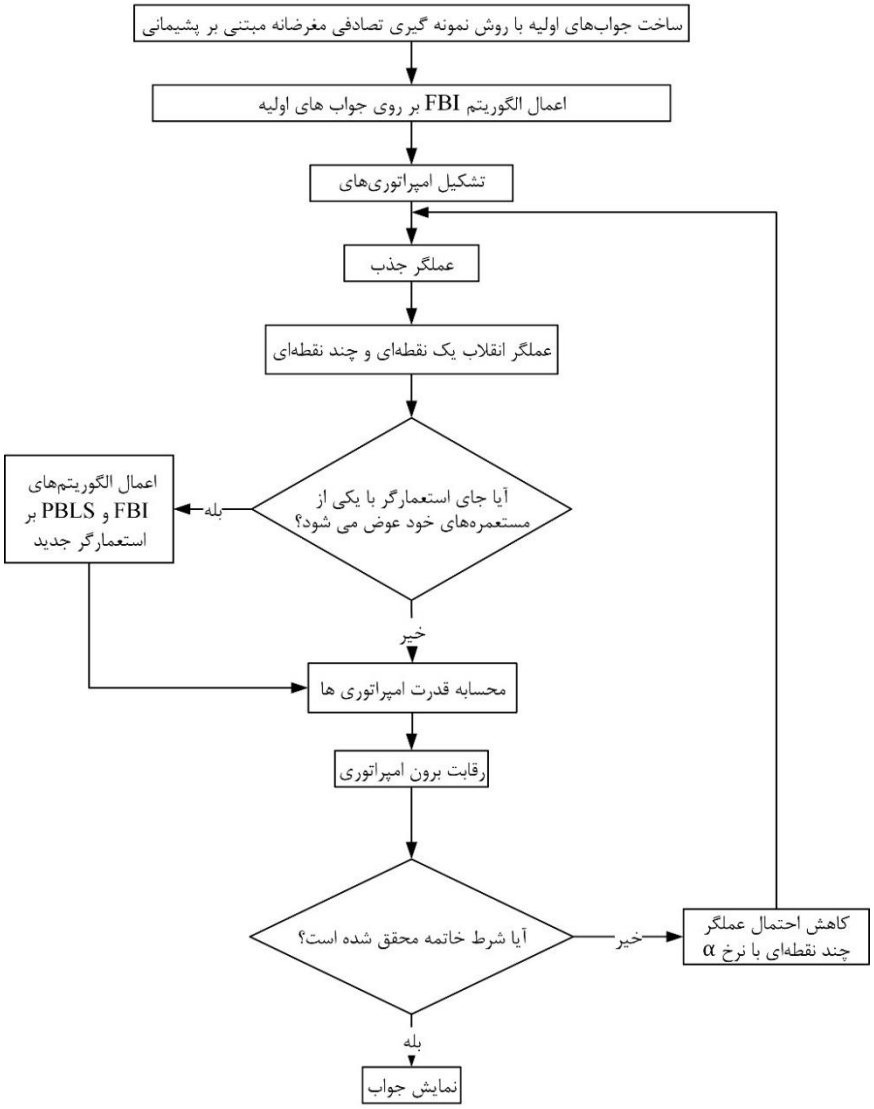
$$T.C_n = Cost(imerialist_n) + \xi mean\{Cost(colonies\ of\ empir \quad (۱)$$

الگوریتم رقابت استعماری پیشنهادی

رویهٔ الگوریتم پیشنهادی ICA به این صورت است که ابتدا جواب‌های اولیه، به روش نمونه‌گیری تصادفی مغرضانهٔ مبتنی‌بر پشیمانی[۱۱] تولید می‌شوند. سپس الگوریتم بهبود پیشرو- پس‌رو[۱۲] (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 به‌عنوان استعمارگر انتخاب می‌شود. سایر جواب‌ها، به‌عنوان کلونی به هر استعمارگر تخصیص داده می‌شود. پس از تشکیل امپراتوری‌ها، هریک از کلونی‌ها به‌وسیلهٔ عملگر تقاطع یکنواخت، به‌سمت استعمارگر خود جذب می‌شود. در مرحلهٔ بعد، عملگر انقلاب یک‌نقطه‌ای 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 مبتنی‌بر جایگشت[۱۳] (PBLS) برای بهبود بیشتر استعمارگر جدید، بر آن اعمال می‌شوند. درنهایت، رقابت میان امپراتوری‌ها برای تصاحب بدترین جواب از بدترین امپراتوری شکل می‌گیرد. رویهٔ این الگوریتم در شکل ۳ مشاهده می‌شود.

کدگذاری و ساخت جواب‌های اولیه

در الگوریتم رقابت استعماری پیشنهادی، جواب‌ها به‌صورت فهرست فعالیت[۱۴] 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 مغرضانهٔ مبتنی‌بر 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 به‌صورت تصادفی انتخاب می‌شود و درون فهرست قرار می‌گیرد. فعالیت i کاندید است، اگر درون فهرست نیمه‌ساخته قرار نگرفته باشد و همهٔ پیش‌نیازهای آن در فهرست نیمه‌ساخته وجود داشته باشد. برای ساخت هر فهرست، یکی از قوانین اولویت LTS، LFT و MTS، به‌صورت تصادفی انتخاب می‌شود. از قانون اولویت انتخاب‌شده، برای تعیین احتمال انتخاب فعالیت‌ها در هر مرحله استفاده می‌شود.



شکل ۳. چارچوب الگوریتم رقابت استعماری پیشنهادی

احتمال انتخاب فعالیت j با اندازهٔ اولویت μ_j به‌وسیلهٔ رابطهٔ ۲ محاسبه می‌شود.

$$\eta_j = \frac{(\mu_j+\varepsilon)^\alpha}{\sum_{j\in D}(\mu_j+\varepsilon)^\alpha} \quad (۲)$$

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

$$\mu_j = max_{i\in D}(LFT_i) - LFT_j + ۱ \quad (۳)$$

$$\mu_j = max_{i\in D}(LST_i) - LST_j + ۱ \quad (۴)$$

$$\mu_j = MTS_j - min_{i\in D}(MTS_i) + ۱ \quad (۵)$$

دلیل انتخاب سه قانون اولویت LFT، LST و MTS این است که به‌وسیلهٔ تولید جواب‌های اولیهٔ متنوع، مناطقی از فضای جواب که احتمال می‌رود جواب بهینه در آنجا قرار داشته باشد، جست‌وجو شوند [۱۳]. علاوه‌براین، کولیش نشان داده است که این سه قانون اولویت، در مقایسه با سایر قوانین عملکرد بهتری دارند. کولیش برای انتخاب فعالیت‌ها، روش نمونه‌گیری تصادفی مغرضانهٔ مبتنی‌بر پشیمانی را با $\alpha = \varepsilon = ۱$ پیشنهاد کرده است. از این‌رو، در این تحقیق نیز این دو پارامتر برابر با ۱ درنظر گرفته شده‌اند [۱۷]. پس از ساخت جواب‌های اولیه، الگوریتم FBI

بر همهٔ جواب‌ها اعمال می‌شود. درنهایت، براساس مقدار تابع هدف، استعمارگران مشخص می‌شوند و کشورها به‌عنوان مستعمره بین آن‌ها تقسیم می‌شوند.

کدگشایی جواب‌ها

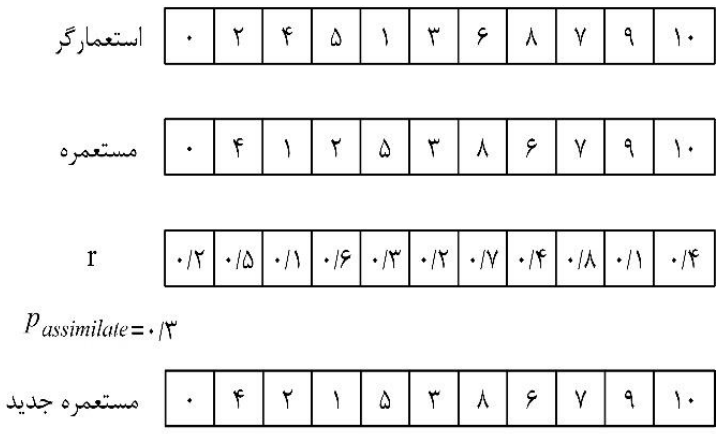
برای رمزگشایی جواب‌ها، دو الگوریتم در ادبیات زمان‌بندی پروژه پیشنهاد شده است. الگوریتم SSGS[15] که براساس فعالیت‌ها و روابط پیش‌نیازی آن‌ها عمل می‌کند. روش دوم، الگوریتم رویهٔ تولید برنامهٔ زمان‌بندی موازی PSGS[16] که نقاط تصمیم‌گیری آن مبتنی‌بر زمان است. تفاوت میان SSGS و PSGS در آن است که زمان‌بندی‌های تولیدشده به‌وسیلهٔ SSGS، زمان‌بندی‌های فعال[17] هستند. درحالی‌که زمان‌بندی‌های تولیدشده به‌وسیلهٔ PSGS بدون تأخیر[18] هستند. در زمان‌بندی‌های بدون تأخیر، زمان بیکاری فقط وقتی در برنامهٔ زمان‌بندی ایجاد می‌شود که فعالیت موجه برای زمان‌بندی وجود نداشته باشد. زمان‌بندی بهینه، همواره بدون تأخیر نیست. از این‌رو، به‌کارگیری PSGS ممکن است به نادیده‌گرفتن جواب بهینه منجر شود؛ بنابراین، زمان‌بندی‌های بدون تأخیر، زیرمجموعهٔ زمان‌بندی‌های فعال هستند. در مقابل، زمان‌بندی‌های فعال حداقل یک زمان‌بندی بهینه دارند [۱۷]؛ بنابراین، در این تحقیق، با توجه به این ویژگی مهم الگوریتم SSGS، از آن برای رمزگشایی جواب‌ها استفاده شده است.

در الگوریتم SSGS، فعالیت‌ها به همان ترتیبی که در فهرست فعالیت‌ها قرار گرفته‌اند، وارد الگوریتم می‌شوند و در نزدیک‌ترین زمان ممکن برای آغاز فعالیت زمان‌بندی می‌شوند؛ به‌طوری‌که پیش‌نیازهای فعالیت درون برنامهٔ زمان‌بندی قرار گرفته باشند و منابع کافی برای اجرای فعالیت وجود داشته باشد.

عملگر جذب

به‌منظور مدل‌سازی سیاست جذب برای مسئلهٔ زمان‌بندی پروژه، از عملگر تقاطع یکنواخت پیشنهادی هارتمان [۷] استفاده شده است. این عملگر به این صورت است که ابتدا فهرستی جدید، به اندازهٔ تعداد فعالیت‌های مسئله، از عددهای تصادفی $r_i \in [0,1]$ تولید می‌شود. سپس باید برای هر جایگاه از جواب جدید تصمیم‌گیری شود. اگر $p_{assimilate} \geq r_i$ باشد، فعالیتی که دارای کمترین اندیس است و پیش‌تر در فهرست جدید قرار نگرفته است، از فهرست فعالیت استعمارگر انتخاب می‌شود و در جایگاه iام فهرست جدید قرار می‌گیرد. 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 می‌شود و در جایگاه iام فهرست جدید قرار می‌گیرد. شکل ۴ چگونگی اجرای این عملگر را نشان می‌دهد.

$p_{assimilate}$، نرخ جذب و از پارامترهای الگوریتم است. جواب‌های جدید ایجادشده به‌وسیلهٔ این عملگر همواره موجه هستند.

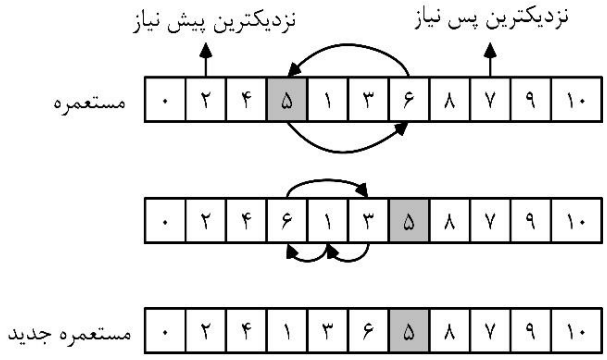


شکل ۴. عملگر جذب

عملگر انقلاب تک‌نقطه‌ای

عملگر انقلاب تک‌نقطه‌ای به این صورت است که از فهرست فعالیت‌های هر کلونی، یک فعالیت به‌صورت تصادفی انتخاب می‌شود. این فعالیت را می‌توان در فاصلهٔ بین نزدیک‌ترین پیش‌نیاز و پس‌نیاز جابه‌جا کرد؛ بنابراین، مکان جدید فعالیت، به‌صورت 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 نحوهٔ کارکرد این عملگر را برای جواب مسئلهٔ شکل ۱ نشان می‌دهد. ابتدا انتخاب فعالیت ۵ به‌صورت تصادفی صورت می‌گیرد. نزدیک‌ترین پیش‌نیاز و پس‌نیاز فعالیت ۵ در این فهرست، به‌ترتیب فعالیت‌های ۲ و ۷ هستند؛ بنابراین، مکان جدید فعالیت ۵ باید از میان مکان فعالیت‌های ۴، ۱، ۳، ۶، ۸ به‌تصادف 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 تولیدشدهٔ جدید همچنان موجه است.



شکل ۵. عملگر انقلاب یک‌نقطه‌ای

عملگر انقلاب چندنقطه‌ای

این عملگر، فهرست فعالیت‌ها را به‌صورت کنترل‌شده به فهرست دیگری تبدیل می‌کند. عملگر چندنقطه‌ای به‌گونه‌ای 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 درنظر گرفته شده است و در هر تکرار، $P_{revolution}$ با نرخ α کم می‌شود.

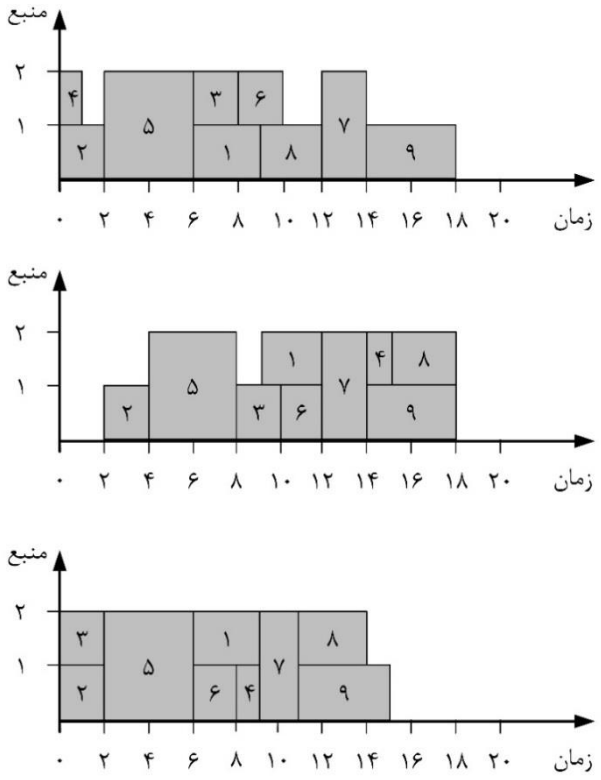
رقابت میان امپراتوری‌ها

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

روش بهبود پیشرو- پس‌رو

الگوریتم FBI روشی مؤثر برای بهبود برنامهٔ زمان‌بندی

است که به‌وسیلهٔ لی و ویلس معرفی شده است [۱۸]. به‌طورکلی در الگوریتم FBI، با به‌کارگیری متناوب SSGS به‌صورت پیشرو و پس‌رو، جواب‌ها تا جایی بهبود پیدا می‌کنند که امکان بهبود بیشتر وجود نداشته باشد. زمان پایان فعالیت‌ها که با محاسبات پیشرو به‌دست آمده است، اولویت فعالیت‌ها را برای محاسبات پس‌رو بعدی تعیین می‌کند. به‌طور مشابه، زمان شروع فعالیت‌ها که با محاسبات پس‌رو به‌دست آمده است، اولویت فعالیت‌ها را برای محاسبات پیشرو بعدی تعیین می‌کند. شکل ۶ یک تکرار از الگوریتم FBI را برای بهبود زمان‌بندی شکل ۲ نشان می‌دهد.



شکل ۶. یک تکرار از الگوریتم FBI [۱۲]

الگوریتم PBLS

الگوریتم PBLS مانند عملگر انقلاب چندنقطه‌ای است؛ با این تفاوت که درصورتی جای فعالیت i با فعالیت $i+1$ عوض می‌شود که فعالیت i پیش‌نیاز فعالیت $i+1$ نباشد و این جابه‌جایی، به بهبود زمان پروژه منجر شود. در این الگوریتم، همهٔ جابه‌جایی‌های مجاز بررسی می‌شوند و درصورت بهبود جواب، این جابه‌جایی حفظ می‌شود.

نتایج محاسباتی

به‌منظور ارزیابی عملکرد الگوریتم پیشنهادی، ۹۶۰ مسئله از مجموعه مسائل PSBLIB انتخاب شده‌اند. مسائل PSPLIB به‌وسیلهٔ نرم‌افزار ProGen تولید شده‌اند و این نرم‌افزار به دست کولیش و اسپریچر طراحی شده است [۱۹]. این مجموعه، ۴۸۰ مسئله با ۳۰ فعالیت و ۴۸۰ مسئله با ۶۰ فعالیت دارد.

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

(۶)

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

$Makespan_i$ مقدار تابع هدف به‌دست‌آمده به‌وسیلهٔ الگوریتم رقابت استعماری برای مسئلهٔ i ام است. LB_i حد پایین مسئلهٔ i ام و R تعداد مسائل است. برای مسائل ۳۰ فعالیتی، جواب بهینه به‌عنوان حد پایین آن‌ها درنظر گرفته شده است، اما از آنجاکه جواب بهینهٔ مسائل مجموعهٔ ۶۰ فعالیتی مشخص نیست، از حد پایان روش مسیر بحرانی[۲۰] برای محاسبهٔ 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.

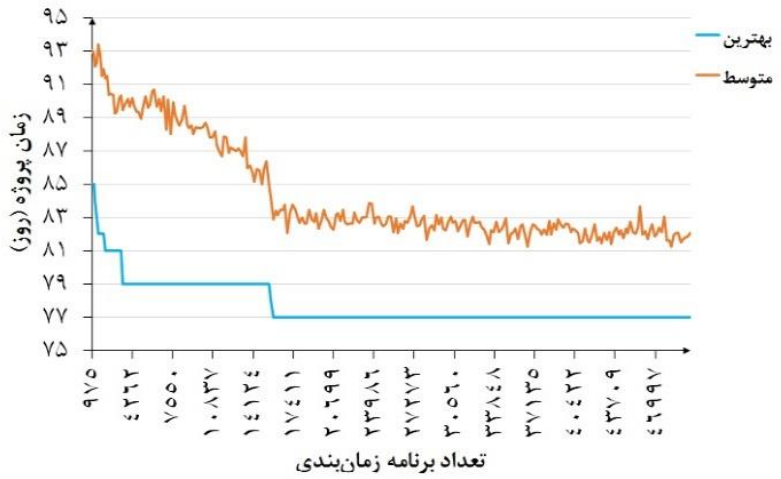
تنظیم پارامترها

برای تنظیم پارامترهای الگوریتم رقابت استعماری، از طراحی آزمایش تاگوچی استفاده شده است. به همین جهت، ۴۸ مسئله از مجموعهٔ ۳۰ فعالیتی و ۴۸ مسئله از مجموعهٔ ۶۰ فعالیتی، به‌صورت 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 (α) و ضریب (ξ) 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 قابل‌مشاهده است. Avg.LB.Dev متغیر پاسخ است و ارزیابی ۵۰۰۰ برنامهٔ زمان‌بندی به‌عنوان شرط توقف الگوریتم درنظر گرفته شده است. با مشاهدهٔ 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.

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
α	۰/۰۰۳	۰/۰۰۵	۰/۰۰۸	۰/۰۱
ξ	۰/۰۱	۰/۰۵	۰/۱	/۱۵



شکل ۷. نمودار همگرایی مسئلهٔ j605_6

به‌منظور بررسی همگرایی الگوریتم ICA با پارامترهای انتخابی، یک مسئله به‌صورت تصادفی انتخاب، و نمودار همگرایی حل این مسائل رسم شده است (شکل ۷). این نمودار، بهترین زمان پروژهٔ یافت‌شده به‌وسیلهٔ الگوریتم ICA و متوسط زمان پروژهٔ همهٔ کشورها را در هر تکرار از الگوریتم نشان می‌دهد. از آنجاکه الگوریتم نسبتاً به‌سرعت همگرا می‌شود، پارامترهای انتخاب‌شده برای الگوریتم مناسب‌اند.

جدول ۲. ترکیب انتخاب‌شده از پارامترها

مجموعه مسائل	N_{pop}	N_{imp}	$P_{assimilate}$	α	ξ
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 می‌دهد. متوسط درصد انحراف از جواب بهینه برای مجموعهٔ ۳۰ فعالیتی، به ازای معیار توقف ارزیابی ۱۰۰۰، ۵۰۰۰ و ۵۰۰۰۰ برنامهٔ زمان‌بندی به‌ترتیب برابر با ۰/۳۰، ۰/۱۳ و ۰/۰۱ درصد است. الگوریتم ICA قادر بوده است جواب بهینهٔ ۴۲۶، ۴۴۴ و ۴۷۴ مسئله از ۴۸۰ مسئله را بیابد.

متوسط درصد انحراف از حد پایین برای مجموعهٔ ۶۰ فعالیتی، به ازای معیار توقف ارزیابی ۱۰۰۰، ۵۰۰۰ و ۵۰۰۰۰ برنامهٔ زمان‌بندی، به‌ترتیب ۱۲/۵۳، ۱۲/۲۰ و ۱۱/۷۴ درصد است. همچنین الگوریتم پیشنهادی ۳۱۵، ۳۵۷ و ۳۶۶ عدد از بهترین جواب‌های یافت‌شده برای ۴۸۰ مسئله را یافته است.

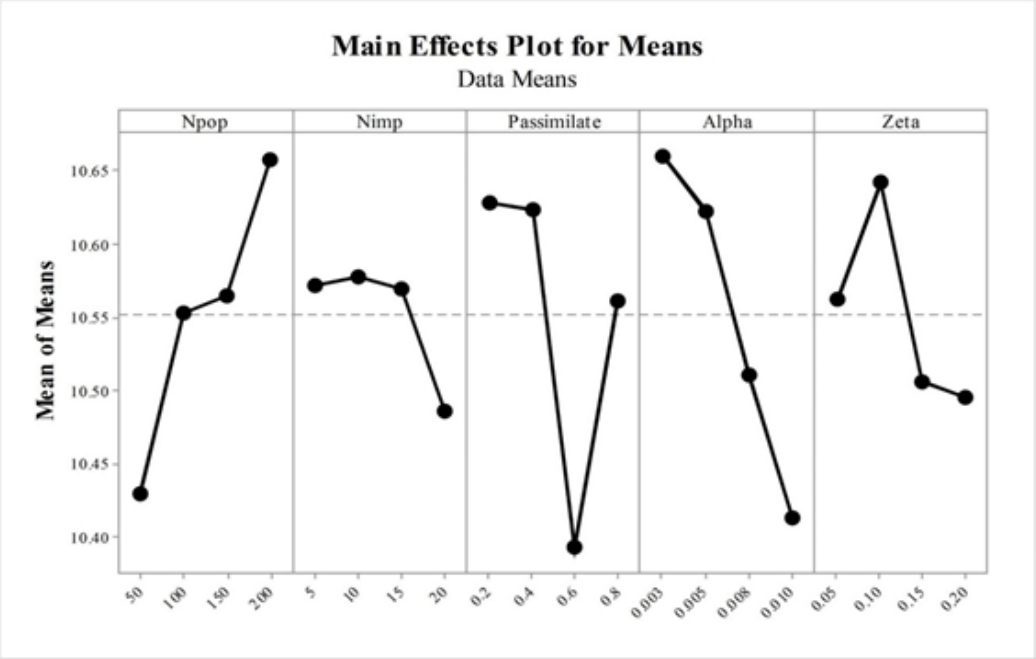
جدول ۳. نتایج الگوریتم ICA پیشنهادی

مجموعه مسائل	معیار توقف (تعداد برنامه‌های زمان‌بندی)		
	۱۰۰۰	۵۰۰۰	۵۰۰۰۰
۳۰ فعالیتی	۰/۳۰	۰/۱۳	۰/۰۱
۶۰ فعالیتی	۱۲/۵۳	۱۲/۲۰	۱۱/۷۴

مقایسهٔ عملکرد الگوریتم پیشنهادی با سایر الگوریتم‌ها

در این قسمت، الگوریتم ICA پیشنهادی با ۱۲ الگوریتم موجود در ادبیات مقایسه شده است. مقادیر به‌دست‌آمده به‌وسیلهٔ این الگوریتم‌ها، برای مجموعه مسائل ۳۰ و ۶۰ فعالیتی در جدول‌های ۵ و ۶ نشان داده شده است.

شکل ۸. روند تغیر پارامترها برای مسائل ۳۰ فعالیتی



شکل ۹. روند تغییر پارامترها برای مسائل ۶۰ فعالیتی

جدول ۴. طبقه‌بندی متعامد طراحی آزمایش‌ها و مقادیر پاسخ

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

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

الگوریتم	تعداد برنامه‌های زمان‌بندی		
	۱۰۰۰	۵۰۰۰	۵۰۰۰۰
الگوریتم پیشنهادی	۰/۳۰	۰/۱۳	۰/۰۱
نمونه‌برداری- انطباقی [۲۰]	۰/۷۴	۰/۵۲	–
ژنتیک [۷]	۱/۰۳	۰/۵۶	۰/۲۳
نمونه‌برداری- انطباقی [۲۱]	۰/۶۵	/۴۴	–
ژنتیک [۸]	۰/۳۸	۰/۲۲	–
تبرید شبیه‌سازی‌شده [۹]	۰/۳۸	۰/۲۳	–
ژنتیک [۲۲]	۰/۷۴	۰/۳۳	۰/۱۶
ژنتیک ترکیبی [۱۰]	۰/۲۷	۰/۰۶	۰/۰۲
زنبورعسل- FBI [۱۱]	۰/۴۲	۰/۱۹	۰/۰۴
نروژنتیک [۲۳]	۰/۱۳	۰/۱۰	–
جهش ترکیبی قورباغه [۱۲]	۰/۳۶	۰/۲۱	۰/۱۸
بهینه‌سازی ازدحام ذرات [۱۳]	۰/۲۲	۰/۰۵	۰/۰۲
بهینه‌سازی چندعاملی [۱۴]	۰/۱۷	۰/۰۶	۰/۰۱

رتبهٔ عملکرد الگوریتم پیشنهادی در مقایسه با سایر الگوریتم‌ها، برای مجموعهٔ ۳۰ فعالیتی و به ازای معیار توقف ارزیابی ۱۰۰۰، ۵۰۰۰ و ۵۰۰۰۰ برنامهٔ زمان‌بندی به‌ترتیب ۴، ۵ و ۱ است.

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

الگوریتم	تعداد برنامه‌های زمان‌بندی		
	۱۰۰۰	۵۰۰۰	۵۰۰۰۰
الگوریتم پیشنهادی	۱۲/۵۳	۱۲/۲۰	۱۱/۷۴
نمونه‌برداری- انطباقی [۲۰]	۱۳/۵۱	۱۳/۰۶	–
ژنتیک [۷]	۱۳/۳۰	۱۲/۷۴	۱۲/۲۶
نمونه‌برداری- انطباقی [۲۱]	۱۲/۹۴	۱۲/۵۸	–
ژنتیک [۸]	۱۲/۲۱	۱۱/۷۰	–
تبرید شبیه‌سازی‌شده [۹]	۱۲/۷۵	۱۱/۹	–
ژنتیک [۲۲]	۱۳/۸	۱۳/۳۱	۱۲/۸۳
ژنتیک ترکیبی [۱۰]	۱۱/۵۶	۱۱/۱۰	۱۰/۷۳
زنبورعسل- FBI [۱۱]	۱۱/۸۹	۱۱/۱۹	۱۰/۸۴
نروژنتیک [۲۳]	۱۱/۵۱	۱۱/۲۹	–
جهش ترکیبی قورباغه [۱۲]	۱۱/۴۴	۱۰/۸۷	۱۰/۶۶
بهینه‌سازی ازدحام ذرات [۱۳]	۱۱/۸۶	۱۱/۱۹	۱۰/۸۵
بهینه‌سازی چندعاملی [۱۴]	۱۱/۶۴	۱۰/۸۴	۱۰/۶۴

در مجموعهٔ ۶۰ فعالیتی، الگوریتم ICA با معیار توقف ارزیابی ۱۰۰۰ برنامهٔ زمان‌بندی هشتمین الگوریتم، با معیار ۵۰۰۰ برنامهٔ زمان‌بندی، نهمین الگوریتم و با معیار توقف ارزیابی ۵۰۰۰۰ برنامهٔ زمان‌بندی، ششمین الگوریتم است.

براساس تعریف کولیش و هارتمان، برای مقایسهٔ الگوریتم‌های فراابتکاری، الگوریتم A، الگوریتم B را مغلوب می‌کند، اگر به ازای همهٔ معیارهای توقف و همهٔ 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، به‌وسیلهٔ 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 مسئلهٔ زمان‌بندی پروژه با محدودیت منابع است. در این الگوریتم، جواب‌ها به‌صورت فهرست فعالیت کدگذاری شده‌اند و برای کدگشایی جواب‌ها، از الگوریتم SSGS استفاده شده است. جواب‌های اولیه با سه قانون اولویت LFT، LST و MTS و به‌وسیلهٔ روش نمونه‌گیری تصادفی مغرضانهٔ مبتنی‌بر پشیمانی 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، به‌وسیلهٔ طراحی آزمایش تاگوچی تنظیم شده‌اند. در انتها، کارایی الگوریتم پیشنهادی، با حل مجموعه مسائل PSPLIB- که شامل ۴۸۰ مسئلهٔ ۳۰ فعالیتی و ۴۸۰ مسئلهٔ ۶۰ فعالیتی است- بررسی شده است. نتایج محاسباتی و مقایسهٔ 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 مسئلهٔ زمان‌بندی پروژه با محدودیت منابع، الگوریتمی کارا به‌شمار می‌رود.

مراجع

1. 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.

2. Hartmann, S. (1997). *Scheduling medical research experiments: an application of project scheduling methods*, Technical Report, University Kiel, Germany.

3. Alba, E. and Francisco Chicano, J. (2007). "Software project management with Gas", *Information Sciences*, Vol. 177, No. 11, PP. 2380–2401.

4. 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.

5. Sprecher, A. (1994). "*Special cases"*, In Resource-constrained project scheduling: Exact methods for the multi-mode case, 1th Ed, PP. 10–18, Springer, Berlin, Germany.

6. Demeulemeester, E. L. and Herroelen, W. S. (2002). *The Resource-Constrained Project Scheduling Problem*, In Project Scheduling: A Research Handbook, 1th Ed, PP. 203–342, Springer, Berlin, Germany.

7. Hartmann, S. (1998). "A competitive genetic algorithm for resource-constrained project scheduling", *Naval Research Logistics*, Vol. 45, No. 6, PP. 733–750.

8. Hartmann, S. (2002). "A self-adapting genetic algorithm for project scheduling under resource constraints", *Naval Research Logistics*, Vol. 49, No. 5, PP. 433–448

9. 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 Drexl, 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