Journal of Soft Computing and Information Technology (JSCIT)
Babol Noshirvani University of Technology, Babol, Iran
Journal Homepage: jscit.nit.ac.ir
Volume 10, Number 3, Autumn 2021, pp. 86-98
Received: 05/01/2020, Revised: 03/07/2021, Accepted: 31/08/2021

Improving the Estimation of Software Development Effort Using the Combination of Cuckoo Search and Particle Swarm Optimization Algorithms

Saeed Samavatian[1] and Keyvan Mohebbi[2*]
1- Department of Computer Engineering, Isfahan (Khorasgan) Branch, Islamic Azad University, Isfahan, Iran.
2*- Department of Electrical and Computer Engineering, Mobarakeh Branch, Islamic Azad University, Mobarakeh, Isfahan, Iran.
[1]saeed.samavatian@yahoo.com and [2*]k.mohebbi@mau.ac.ir

Corresponding author address: Keyvan Mohebbi, Department of Electrical and Computer Engineering, Mobarakeh Branch, Islamic Azad University, Mobarakeh, Isfahan, Iran.

Abstract- Accurate estimation of required effort for software development has an important role in success of such projects. So far, a lot of research work has been conducted to estimate the effort, but improving the precision of this calculation is still a challenge. In this paper, an approach is proposed based on the metaheuristic algorithms to solve this challenge. The procedure is as follows. First, the Cuckoo Search algorithm is used in order to select the correct software features in estimating effort. Then, the results are further analyzed by Particle Swarm Optimization algorithm. The idea is that the sequential application of these algorithms has led to more accurate search of the problem space and possibility of achieving the global optimum, i.e. the best features is increased. Finally, the selected features are used as the input parameters of the COCOMO II post-architecture model and the effort is estimated. The proposed approach is evaluated on two datasets of COCOMO 81 and COCOMO NASA and in order to its evaluation, two metrics, namely the median magnitude of relative error and the percentage of prediction are used. The results obtained from the experiments of this approach and their comparison to the results of the previous works show that on the COCOMO 81, the value of the median magnitude of relative error decreased by 0.177 and the percentage of prediction, for the three values of 25, 30 and 40 percent, increased by 7.87%, 8.04% and 8.66%, respectively. Furthermore, on the COCOMO NASA, the value of the median magnitude of relative error decreased by 0.151 and the percentage of prediction, for the three values of 25, 30 and 40 percent, increased by 7.55%, 7.98% and 8.11%, respectively.

Keywords- Effort Estimation, Software Development, Cuckoo Search, Particle Swarm Optimization.

مجله علمی پژوهشی رایانش نرم و فناوری اطلاعات
دانشگاه صنعتی نوشیروانی بابل
صفحه مجله: jscit.nit.ac.ir
جلد ۱۰، شماره ۳، پاییز ۱۴۰۰، صفحه ۸۶-۹۸
دریافت: ۱۳۹۹/۱۰/۱۶، بازنگری: ۱۴۰۰/۰۴/۱۲، پذیرش: ۱۴۰۰/۰۶/۰۹

بهبود تخمین تلاش توسعه نرم‌افزار با استفاده از ترکیب الگوریتم‌های جستجوی فاخته و بهینه‌سازی ازدحام ذرات

سعید سماواتیان[۱] و کیوان محبی*[۲]
۱- گروه کامپیوتر، واحد اصفهان (خوراسگان)، دانشگاه آزاد اسلامی، اصفهان، ایران.
*۲- گروه برق و کامپیوتر، واحد مبارکه، دانشگاه آزاد اسلامی، مبارکه، اصفهان، ایران.
[1]saeed.samavatian@yahoo.com , [2*]k.mohebbi@mau.ac.ir

* نشانی نویسنده مسئول: کیوان محبی، گروه برق و کامپیوتر، واحد مبارکه، دانشگاه آزاد اسلامی، مبارکه، اصفهان، ایران.

چکیده- تخمین صحیح تلاش لازم برای توسعه نرم‌افزار، نقش مهمی در موفقیت این قبیل پروژه‌ها دارد. تاکنون پژوهش‌های متعددی برای تخمین تلاش انجام شده است، لیکن بهبود دقت این محاسبه هنوز از چالش‌های مطرح است. در این مقاله، راهکاری مبتنی بر الگوریتم‌های فراابتکاری برای حل این چالش ارائه شده است. روش کار به این صورت است که ابتدا از الگوریتم جستجوی فاخته به منظور انتخاب صحیح ویژگی‌های نرم‌افزاری مطرح در تخمین تلاش استفاده می‌شود. سپس جواب‌های به دست آمده با استفاده از الگوریتم بهینه‌سازی ازدحام ذرات بیشتر مورد واکاوی قرار می‌گیرد. ایده این کار آن است که اجرای متوالی الگوریتم‌های مذکور باعث جستجوی دقیق‌تر فضای مسأله شده و امکان دسترسی به بهینه سراسری، یعنی ویژگی‌های بهینه را افزایش دهد. در نهایت، ویژگی‌های انتخاب شده به عنوان پارامترهای ورودی مدل پسا معماری کوکومو۲ مورد استفاده قرار گرفته و تلاش لازم، محاسبه می‌شود. راهکار پیشنهادی بر روی دو مجموعه داده کوکومو۸۱ و کوکوموناسا مورد بررسی قرار گرفته و به‌منظور ارزیابی آن از دو معیار متوسط شدت خطای نسبی و درصد پیش‌بینی استفاده شده است. نتایج به‌دست آمده از آزمایش‌های این راهکار و مقایسه آن با پژوهش‌های پیشین نشان می‌دهد که در کوکومو۸۱، مقدار متوسط شدت خطای نسبی به اندازه ۰/۱۷۷ کاهش یافته و درصد پیش‌بینی به ترتیب در سه حالت ۲۵، ۳۰ و ۴۰ درصد، به اندازه ۷/۸۷٪، ۸/۰۴٪ و ۸/۶۶٪ افزایش یافته است. همچنین در کوکوموناسا، مقدار متوسط شدت خطای نسبی به اندازه ۰/۱۵۱ کاهش یافته و درصد پیش‌بینی به ترتیب در سه حالت ۲۵، ۳۰ و ۴۰ درصد، به اندازه ۷/۵۵٪، ۷/۹۸٪ و ۸/۱۱٪ افزایش یافته است.

واژه‌های کلیدی: تخمین تلاش، توسعه نرم‌افزار، جستجوی فاخته، بهینه‌سازی ازدحام ذرات.

۱- مقدمه

توسعه کیفی نرم‌افزار یک امر حیاتی و ضروری برای مدیران پروژه می‌باشد و اینکه در سال‌های اخیر، تخمین تلاش توسعه نرم‌افزار[۱] به عنوان یکی از اهداف اصلی مدیران و تیم توسعه نرم‌افزار تبدیل شده است [۱]. مرور ادبیات نشان می‌دهد، که به‌منظور دستیابی به نرم‌افزاری با کیفیت بالا که در زمان مقرر و با هزینه‌ی تعیین شده توسعه یابد، تخمین تلاش نقش مهمی را ایفا می‌کند [۵-۲]. اگر در مراحل اولیه چرخه حیات توسعه نرم‌افزار[۲]، تخمین تلاش با اشتباه همراه باشد آنگاه اتفاقات ناگوار، پُرهزینه و جبران‌ناپذیری به دنبال خواهد داشت؛ بنابراین شرکت‌های نرم‌افزاری به‌منظور توسعه نرم‌افزارهای کارآمد و با کیفیت نیازمند تخمین دقیق تلاش پیش از شروع پروژه می‌باشند. تخمین دقیق تلاش در توسعه‌ی یک سیستم نرم‌افزاری باعث می‌شود تا مدیران پروژه نرم‌افزاری بتوانند تصمیمات قدرتمندی اتخاذ نمایند و نسبت به نیازمندی‌های خود در زمینه‌ی میزان هزینه، زمان و نیروی انسانی برای تولید یک محصول موفق آگاهی پیدا کنند [۶]. در حقیقت تخمین تلاش در ابتدای فرآیند توسعه‌ی نرم‌افزار اغلب با دقت کمی همراه است، زیرا اطلاعات بسیار کمی از پروژه در اختیار می‌باشد [۷]. این موضوع باعث می‌شود تا برنامه‌ریزی پروژه با ریسک همراه شود که افزایش ریسک و عدم

اطمینان از تخمین اولیه را به دنبال خواهد داشت. همچنین ذکر این نکته ضروری است که نتایج به‌دست آمده از تخمین‌ها می‌تواند به‌منظور ارائه پیشنهادها، انجام مذاکرات برای عقد قرارداد، زمان‌بندی و نظارت استفاده گردد. لذا با توجه به تمامی موارد مطرح شده می‌توان به این نتیجه رسید که تخمین دقیق تلاش در آغاز پروژه نرم‌افزاری یک مرحله‌ی مهم و ضروری برای مدیران و مشتریان پروژه می‌باشد.

تاکنون روش‌های متفاوتی در زمینه‌ی تخمین تلاش ارائه شده است [۴, ۵, ۸, ۹]. از سوی دیگر، عوامل متفاوتی مانند پیشرفت سریع سیستم‌عامل‌ها و سخت‌افزارها، تغییر نیازهای مشتری در طول پروژه و تغییر مداوم خصوصیات پروژه‌های نرم‌افزاری بر عملکرد و کارآیی نرم‌افزار تأثیرگذار می‌باشند؛ بنابراین، انتخاب بهترین و مناسب‌ترین روش برای تخمین دقیق تلاش به یک چالش عمده در زمینه توسعه‌ی سیستم‌های نرم‌افزاری تبدیل شده است.

مرور ادبیات نشان می‌دهد که دو فاکتور بسیار مهم در زمینه بهبود کارآیی روش‌های تخمین تلاش توسعه نرم‌افزار وجود دارد که عبارت‌اند از: کاهش مقدار متوسط شدت خطای نسبی[۳] و افزایش میزان درصد پیش‌بینی[۴]. همچنین در اغلب پژوهش‌های پیشین، از انتخاب صحیح و مناسب ویژگی‌های نرم‌افزاری به‌عنوان یک چالش عمده اشاره شده است [۱, ۱۰, ۱۱].

تاکنون پژوهش‌های زیادی در زمینه تخمین تلاش توسعه نرم‌افزار در حوزه‌های کاربردی متفاوت انجام گرفته است اما مرور منابع اخیر نشان می‌دهد که توجه بسیاری از پژوهشگران به‌سوی استفاده از الگوریتم‌های فراابتکاری و بهینه‌سازی در این زمینه به‌منظور افزایش دقت و بهبود کارآیی معطوف شده است [۱۵-۱۲].

از جمله در [۲]، که از جستجوی فاخته[۵] برای کشف بهترین پارامترهای ممکن مدل کوکومو۲ و سپس ترکیب با شبکه عصبی مصنوعی[۶] به منظور پیش‌بینی بهتر تلاش توسعه‌ی نرم‌افزار استفاده شده است. جستجوی فاخته الهام گرفته از طبیعت بر اساس تولید مثل انگلی فاخته‌ها و یکی از الگوریتم‌های فراابتکاری جدید مبتنی بر جمعیت برای بهینه‌سازی است که تابع هدف[۷] را قادر می‌سازد تا به دنبال جواب بهینه از فضای راه حل باشد [۱۶].

مزیت عمده جستجوی فاخته نسبت به سایر روش‌های فراابتکاری آن است که به جای گام‌های تصادفی ساده استفاده شده در دیگر روش‌ها مثل الگوریتم ژنتیک[۸]، از پرواز لوی[۹] به عنوان یکی از مؤثرترین روش‌های گام‌های تصادفی استفاده نموده و عملکرد بهتری ارائه می‌کند. بعلاوه، شباهت بین جواب‌ها منجر به نتایج جدید بهتری می‌شود که اساسا نسلی متناسب با برازش با یک قابلیت ترکیب خوب است [۱۷, ۱۸]. بنابراین اگرچه ارزیابی مدل پیشنهادی در [۲] نشان دهنده مؤثر بودن آن و نتایج چشم‌گیری است، لیکن استفاده از الگوریتم جستجوی فاخته برای مسائل با ابعاد بالا سبب می‌شود که به جای بهینه سراسری[۱۰]، بهینه محلی[۱۱] به عنوان خروجی مشخص شود که این امر سبب پایین آمدن دقت این الگوریتم خواهد شد. تحقیق حاضر با در نظر گرفتن [۲] به عنوان مقاله پایه، قصد رفع محدودیت آن و در نتیجه افزایش دقت تخمین تلاش توسعه نرم‌افزار را دارد.

برای رفع این محدودیت می‌توان از ترکیب آن با الگوریتم‌های بهینه-سازی استفاده نمود. یکی از این الگوریتم‌ها، بهینه‌سازی ازدحام ذرات[۱۲] می‌باشد که استفاده از آن به دلیل سادگی، پایین بودن هزینه‌ی محاسبات و همچنین اثرگذاری بر روی محدوده وسیعی از کاربردها، افزایش یافته است [۱۹]. مزایای این الگوریتم نسبت به سایر الگوریتم‌های بهینه‌سازی را می‌توان به صورت زیر خلاصه نمود:

بهره‌مندی از حافظه: الگوریتم بهینه‌سازی ازدحام ذرات حاوی حافظه است، به این صورت که همه‌ی ذرات راه‌حل‌های خوب را نگهداری می‌کنند.

همکاری و اشتراک‌گذاری اطلاعات بین ذرات: در این الگوریتم، موقعیت هر عضو جامعه بر اساس تجربیات شخصی و تجربیات کل جامعه تغییر می‌کند.

انعطاف‌پذیری بهتر در برابر مشکل بهینه محلی: الگوریتم بهینه‌سازی ازدحام ذرات در مقایسه با سایر استراتژی‌های بهینه‌سازی، با استفاده از ذرات ازدحام کننده در مقابل مشکل بهینه محلی، انعطاف بیشتری از خود نشان می‌دهد.

راحتی پیاده‌سازی و اجرا: این الگوریتم ساده‌تر از الگوریتم ژنتیک، الگوریتم بهینه‌سازی کلونی مورچه‌ها[۱۳] و سایر الگوریتم‌های مبتنی بر جمعیت می‌باشد. علاوه بر این، مقداردهی اولیه جمعیت در بکارگیری از این الگوریتم، نسبت به سایر الگوریتم‌های بهینه‌سازی هوشمند ساده‌تر می‌باشد.

علاوه بر ویژگی‌های فوق، در تحقیقات مشابه مختلف نشان داده شده که استفاده از الگوریتم بهینه‌سازی ازدحام ذرات می‌تواند معیارهای ارزیابی را تا حد مطلوبی بهبود بخشد [۲۳-۲۰].

از این رو، این مقاله یک روش ترکیبی مبتنی بر الگوریتم‌های جستجوی فاخته و بهینه‌سازی ازدحام ذرات را به‌منظور بهره‌مندی از مزایای هر دو الگوریتم جهت دستیابی به بهینه‌ی سراسری و تخمین دقیق‌تر تلاش توسعه نرم‌افزار پیشنهاد داده است. در راستای این هدف، روش پیشنهادی با به‌کارگیری مدل ترکیبی مذکور، به ترتیب قصد کاهش و افزایش دو معیار ارزیابی مهم در این زمینه با نام‌های متوسط شدت خطای نسبی و میزان درصد پیش‌بینی را دارد.

در ادامه، ساختار مقاله به شرح زیر سازمان‌دهی شده است: بخش دوم به مرور پیشینه پژوهش می‌پردازد. بخش سوم روش پیشنهادی برای تخمین تلاش توسعه نرم‌افزار را معرفی می‌کند. در بخش چهارم، نتایج آزمایش‌ها و ارزیابی کارآیی روش پیشنهادی بر روی مجموعه داده‌های کوکومو۸۱[۱۴] و کوکوموناسا[۱۵] ارائه می‌شود. بخش پنجم به مقایسه روش پیشنهادی با سایر روش‌ها می‌پردازد. در بخش ششم نتیجه‌گیری مقاله انجام می‌گیرد و پیشنهاداتی برای کارهای آتی ارائه می‌شود.

۲- مروری بر پیشینه پژوهش

تاکنون کارهای متفاوتی در زمینه تخمین تلاش توسعه نرم‌افزار انجام شده است که در ادامه به نمونه‌هایی از آن‌ها اشاره می‌شود.

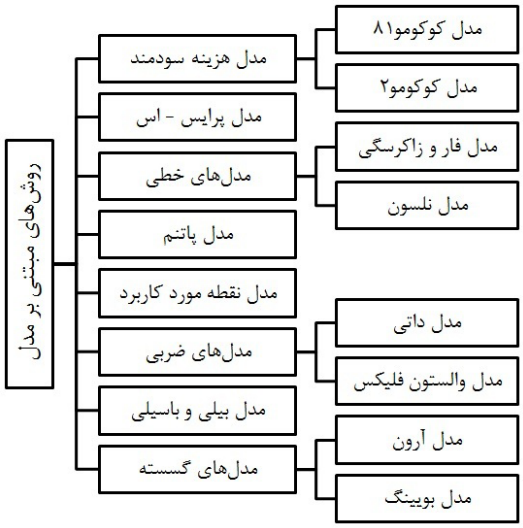
کوماری و پوشکار یک مدل ترکیبی متشکل از الگوریتم جستجوی فاخته و شبکه عصبی مصنوعی را با هدف افزایش دقت تخمین تلاش توسعه نرم‌افزار ارائه نمودند [۲]. در این پژوهش روش پیشنهادی بر روی مجموعه داده‌های نرم‌افزاری کوکومو۸۱ و کوکوموناسا مورد ارزیابی قرار گرفته است. آن‌ها برای ارزیابی، عملکرد معیار متوسط شدت خطای نسبی و سه نوع مختلف درصد پیش‌بینی را در نظر گرفتند. نتایج به‌دست آمده از روش پیشنهادی در مقایسه با سایر مدل‌های موجود نشان می‌دهد که در هر دو مجموعه داده، متوسط شدت خطای نسبی کاهش یافته و درصد پیش‌بینی افزایش پیدا کرده است.

پادماجا و هریتا یک روش فرااکتشافی[۱۶] مانند الگوریتم خفاش[۱۷] را با هدف بهبود دقت تخمین تلاش توسعه نرم‌افزار پیشنهاد نمودند [۳]. در این پژوهش کارآیی مدل پیشنهاد شده بر روی مجموعه داده کِمِرر[۱۸] آزمایش شده است. نتیجه به‌دست آمده با استفاده از مدل پیشنهادی در مقایسه با مدل‌های دیگر نظیر تحلیل رابطه خاکستری[۱۹] برای به حداقل رساندن نرخ خطا نتایج بهتری را نشان داده است.

خُوات و همکاران از الگوریتم کلونی زنبور عسل مصنوعی[۲۰] جهت‌دار به‌منظور تنظیم مقادیر پارامترهای مدل‌های تخمین تلاش توسعه نرم‌افزار بر اساس تلاش واقعی استفاده کرده‌اند [۱۳]. این روش بر روی مجموعه داده‌های نرم‌افزاری ناسا آزمایش شد. نویسندگان معتقدند که عدم وجود همگرایی زودرس[۲۱] در دوره جستجوی بعدی، بهینه‌سازی پارامترهای مدل‌های تخمین تلاش و افزایش دقت تخمین تلاش نرم‌افزار از مزایای روش پیشنهادی آن‌ها می‌باشد.

گوتام و سینگ روش‌های تخمین تلاش توسعه نرم‌افزار را در چهار گروه دسته‌بندی کرده‌اند [۲۴]. بر اساس دسته‌بندی پیشنهادی نویسندگان در این مرجع، روش‌های تخمین تلاش توسعه نرم‌افزار در گروه‌های "مبتنی بر مدل"، "مبتنی بر تشابه"، "مبتنی بر محاسبات نرم" و "ترکیبی" قرار گرفته است.

در پژوهش انجام‌شده توسط مِنزیس و همکاران، یک دسته‌بندی برای روش‌های مبتنی بر مدل ارائه شده است [۲۵]. سپس مزایا و معایب این دسته از روش‌ها مشخص شده است. این دسته‌بندی در شکل ۱ نشان داده شده است.



شکل ۱: دسته‌بندی روش‌های مبتنی بر مدل [۲۵]

بیرانوند و همکاران یک مقایسه میان روش‌های مبتنی بر تشابه انجام داده‌اند[۲۶]. سپس مزایا و معایب این گروه از روش‌ها را ارائه نموده‌اند. آن‌ها در جمع‌بندی پژوهش خود، انعطاف‌پذیری بالا، به‌کارگیری در پروژه‌های پیچیده و عدم نیاز به اطلاعات کامل در ابتدای پروژه را به عنوان مزایای این روش‌ها ذکر کرده‌اند. همچنین آن‌ها معتقدند که فرآیند تخمین در این روش‌ها زمان‌بر می‌باشد.

شاه و همکاران با هدف بهبود دقت تخمین تلاش توسعه نرم‌افزار، یک روش تخمین مبتنی بر تشابه را با استفاده از الگوریتم کلونی زنبور عسل مصنوعی پیشنهاد نمودند [۲۷]. روش پیشنهادی آن‌ها در یک محیط دو مرحله‌ای آزمایش و آموزش کار می‌کند، به این‌صورت که در مرحله آموزش مناسب‌ترین ویژگی‌های الگوریتم کلونی زنبور عسل مصنوعی محاسبه می‌شوند. سپس در مرحله آزمایش دقت تخمین تلاش مورد ارزیابی قرار می‌گیرد.

مصطفی و همکاران یک مدل جنگل تصادفی[۲۲] را طراحی کردند که به طور تجربی با تغییر مقادیر پارامترهای اصلی آن بهینه‌سازی شده بود [۲۸]. آن‌ها این مدل را با هدف بهبود دقت تخمین تلاش پروژه‌های نرم‌افزاری ارائه نمودند. در این پژوهش عملکرد مدل جنگل تصادفی بهینه شده، با درخت رگرسیون[۲۳] کلاسیک مقایسه شده است. نویسندگان معتقدند که مدل جنگل تصادفی بهینه

عملکرد بهتری نسبت به مدل رگرسیون درختی در همه‌ی معیارهای ارزیابی دارد.

اِزغاری و زاهی از روش قیاس فازی[۲۴] با هدف بهبود دقت تخمین تلاش توسعه نرم‌افزار استفاده نمودند [۲۹]. در این مرجع روش پیشنهادی بر روی ۱۳ مجموعه داده نرم‌افزاری آزمایش شده است و سپس نتایج به‌دست آمده از روش پیشنهادی با مدل‌های موجود در سایر ادبیات‌ها مقایسه شده است. نویسندگان معتقدند که روش پیشنهادی آن‌ها نسبت به سایر روش‌ها به طور قابل توجهی عملکرد و دقت بالاتری دارد.

زارع و همکاران روشی برای به‌روزرسانی بهینه ضریب تخمین تلاش بر روی مجموعه داده کوکومو‌ناسا با استفاده از الگوریتم‌های ژنتیک و بهینه‌سازی ازدحام ذرات ارائه کرده‌اند [۳۰]. نویسندگان معتقدند که ضریب بهینه ناشی از روش پیشنهادی آن‌ها منجر به کاهش مقادیر خطا و افزایش دقت تخمین تلاش می‌شود.

به‌عنوان نتیجه‌گیری از مرور پژوهش‌های پیشین می‌توان گفت علیرغم پیشرفت‌های صورت گرفته در مورد تخمین تلاش توسعه نرم‌افزار که گاهی با پیچیدگی زمانی بالا نیز همراه بوده است، دقت محاسبات روش‌های موجود ایده‌آل نبوده و بهبود دقت هنوز یک چالش می‌باشد و جای کار دارد.

۳- روش پیشنهادی

مدل مورد استفاده در این مقاله به‌منظور تخمین تلاش، مدل پَسا معماری[۲۵] کوکومو۲ می‌باشد. این مدل، از ویژگی‌های نرم‌افزاری که توسط روش پیشنهادی انتخاب شده به‌عنوان پارامترهای ورودی خود، جهت تخمین دقیق‌تر تلاش توسعه نرم‌افزار استفاده می‌کند. هدف این بخش تشریح جزئیات روش پیشنهادی برای تخمین دقیق تلاش توسعه نرم‌افزار با ترکیب الگوریتم‌های جستجوی فاخته و بهینه‌سازی ازدحام ذرات می‌باشد. برای دستیابی به این هدف، این بخش در دو زیر بخش "ضرورت ارائه روش پیشنهادی" و "طرح کلی روش پیشنهادی" سازمان‌دهی شده است.

۳-۱- ضرورت ارائه روش پیشنهادی

در این مقاله سه فاکتور مهم به‌عنوان ضروریات در نظر گرفته شده است که این ضروریات در شکل ۲ نمایش داده شده است.

شکل ۲: ضروریات ارائه روش پیشنهادی

این سه ضرورت را می‌توان به شکل زیر بیان نمود:

- اولین چالش در این زمینه انتخاب صحیح ویژگی‌های نرم‌افزاری می‌باشد، زیرا استفاده از ویژگی‌های صحیح باعث افزایش میزان دقت تخمین تلاش می‌شود.
- برای اینکه مشخص شود که کدام دسته از ویژگی‌ها را می‌توان به‌عنوان ویژگی‌های مفید در نظر گرفت، باید معیاری برای ارزیابی این ویژگی‌ها انتخاب نمود تا بتوان از این طریق به سنجش ویژگی‌های انتخابی پرداخت.
- یکی از نکات بسیار مهم در این زمینه استفاده از محاسبات عددی می‌باشد به‌نحوی که بتوان از طریق آن میزان برازندگی ویژگی‌ها را مشخص نمود. لذا ارائه یک مدل ریاضی کارا بسیار مهم می‌باشد.

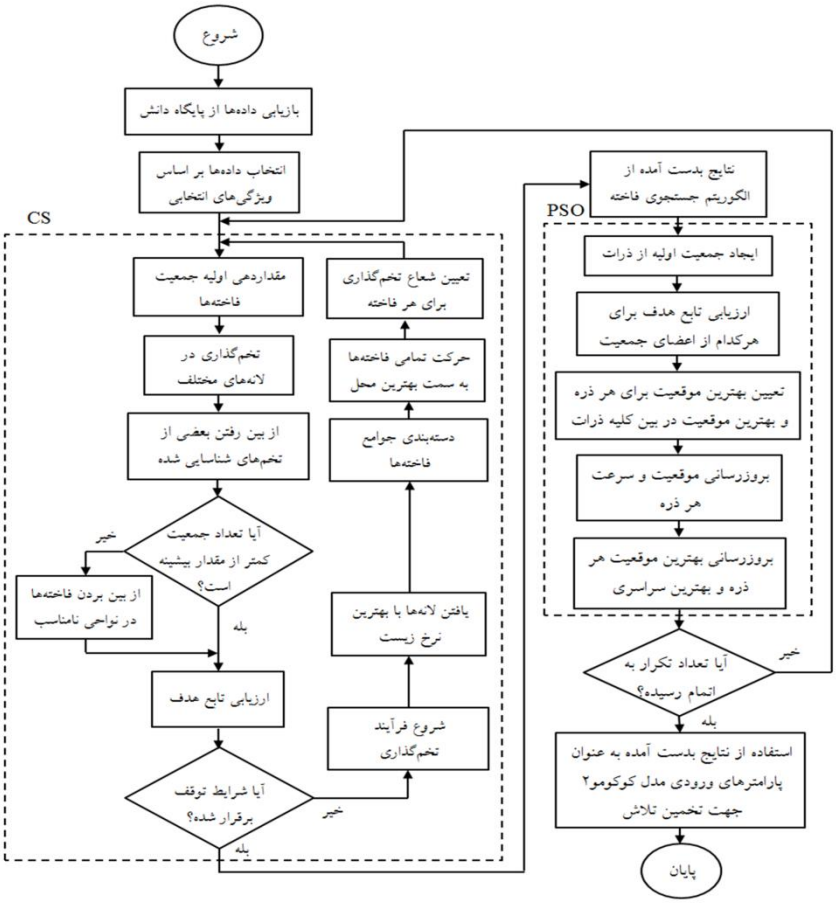
بر اساس این ضروریات در ادامه طرح کلی روش پیشنهادی در قالب فلوچارت ارائه می‌شود.

۳-۲- طرح کلی روش پیشنهادی

در روش پیشنهادی به‌منظور بهبود دقت تخمین تلاش و همچنین انتخاب ویژگی‌ها از ترکیب الگوریتم‌های جستجوی فاخته و بهینه‌سازی ازدحام ذرات استفاده شده است. در واقع از الگوریتم بهینه‌سازی ازدحام ذرات، برای بهبود فرآیند الگوریتم جستجوی فاخته استفاده خواهد شد. طرح کلی روش پیشنهادی در قالب یک فلوچارت در شکل ۳ نمایش داده می‌شود.

مراحل چارچوب روش پیشنهادی به‌صورت زیر شرح داده می‌شوند:

(الف) بازیابی داده‌ها از پایگاه دانش اولین مرحله در این الگوریتم می‌باشد. در این مرحله داده‌های مورد استفاده در برنامه درون ماتریس‌هایی قرار داده می‌شوند تا بتوان در طول برنامه از این ماتریس‌ها برای به‌دست آوردن نتایج و پردازش بر روی داده‌ها استفاده نمود.



شکل ۳: فلوچارت روش پیشنهادی

(ب) مرحله دوم، انتخاب لانه بر اساس ویژگی‌های انتخابی می‌باشد. این مرحله اولین قدم برای پیدا نمودن بهترین ویژگی‌ها می‌باشد. در این مرحله ابتدا به‌صورت تصادفی ویژگی‌هایی انتخاب می‌شوند تا در ادامه الگوریتم، بهینه‌سازی شود و سپس الگوریتم بر اساس تابع برازندگی[۲۶] به سمت بهترین مقدار حرکت نماید. از آنجا که نیاز است تا جواب‌های بدست آمده توسط روش پیشنهادی مورد ارزیابی قرار بگیرند، برای بررسی میزان درستی این جواب‌ها از مفهوم تابع برازندگی یا تابع هدف استفاده می‌شود.

(ج) مرحله سوم، انجام مراحل الگوریتم جستجوی فاخته بر روی ماتریس‌های اولیه می‌باشد. در این مرحله، الگوریتم جستجوی فاخته بر اساس عملگرهای خود بر روی ماتریس‌های اولیه عملیات انجام می‌دهد تا بر اساس تابع برازندگی بهترین ویژگی‌ها به‌دست آیند. حال اگر نتایج به‌دست‌آمده مطلوب باشند، از نتایج به‌دست‌آمده به‌عنوان ورودی الگوریتم بهینه‌سازی ازدحام ذرات استفاده می‌شود، در غیر این صورت فرآیند الگوریتم جستجوی فاخته مجدداً تکرار می‌شود تا الگوریتم با شرایط مناسب‌تری انجام گیرد.

(د) همگرایی نامناسب یک چالش عمده در الگوریتم جستجوی فاخته می‌باشد. این موضوع به‌خصوص برای مسائلی که ابعاد آن‌ها بالا می‌باشد بیشتر نمود می‌کند. در واقع استفاده از الگوریتم جستجوی فاخته برای مسائل با ابعاد بالا سبب می‌شود که الگوریتم در بهینه محلی گیر بیفتد و به‌جای بهینه سراسری، بهینه محلی به‌عنوان خروجی مشخص شود.

این مسأله منجر به بالا رفتن خطای تشخیص این الگوریتم و در نهایت کاهش دقت آن می‌شود؛ بنابراین در این مقاله ترکیب الگوریتم جستجوی فاخته با الگوریتم‌های جستجوی سریع مانند الگوریتم بهینه‌سازی ازدحام ذرات پیشنهاد شده است. از این رو در مرحله چهارم، نتایج به‌دست آمده توسط الگوریتم جستجوی فاخته در قسمت ورودی الگوریتم بهینه‌سازی ازدحام ذرات قرار می‌گیرد. در این مرحله عملیات دیگری بر روی داده‌ها با استفاده از الگوریتم بهینه‌سازی ازدحام ذرات انجام می‌گیرد تا نتایج بهتری حاصل شود. در این صورت فضای جستجو الگوریتم جستجوی فاخته افزایش می‌یابد. حال اگر تعداد تکرار کافی باشد نتایج بهینه موردنظر به‌دست می‌آیند، در غیر این صورت به مرحله اول الگوریتم جستجوی فاخته، بخش مقداردهی اولیه جمعیت فاخته‌ها برمی‌گردد تا با تکرار مراحل، بهینگی مورد نظر حاصل شود.

Algorithm 1: Estimation Software Development Effort

1. **begin**
2. **repeat**
3. Objective function $f(x)$
4. Generate initial population of n host nest
5. Evaluate fitness and rank eggs
6. **while** (t > MaxGeneration) or Stop criterion
7. $t = t + 1$
8. Get a cuckoo randomly/generate new solution by levy flights
9. Evaluate quality/fitness, F_i
10. Choose a random nest j
11. **If** ($F_i > F_j$)
15. Replace j by the new solution
16. **end if**
17. Worst nest is abandoned with probability P_a and new nest is built
18. Evaluate fitness and rank the solutions and fined current best
19. Change nest with PSO
20. $v_{i,d} = wv_{i,d} + C_1 \times R(0,1) \times [pb_{i,d} - x_{i,d}] + C_2 \times R(0,1) \times [gb_d - x_{i,d}]$
21. $x_{i,d} = x_{i,d} + v_{i,d}$
22. Evaluate fitness and if new nest is better replace with old nest
23. **end while**
24. $r = r + 1$
25. **until** $r <= Max\_Iterations$
26. Post process results and visualization
27. Properties selected by these two algorithms as input parameters of
28. COCOMO II post-architecture model
29. Effort = A × (results)^{B} × results
30. **end**

شکل ۴: شبه کد روش پیشنهادی

در مرحله‌ی آخر، از نتایج به‌دست آمده توسط الگوریتم‌های جستجوی فاخته و بهینه‌سازی ازدحام ذرات به‌عنوان پارامترهای ورودی مدل پسا معماری کوکومو۲ استفاده می‌شود تا در این صورت تخمین تلاش انجام گیرد. در ادامه، شبه کد روش پیشنهادی در شکل ۴ نشان داده شده است.

بر اساس این شبه کد، هر لانه (داده) به‌صورت تصادفی ایجاد می‌شود. در واقع هر لانه، ماتریس انتخاب ویژگی‌ها می‌باشد به‌نحوی که در قدم اول توسط الگوریتم جستجوی فاخته انتخاب می‌شود. در ادامه الگوریتم پیشنهادی، بر اساس حلقه‌ی ایجاد شده آغاز می‌شود. در این حلقه ابتدا لانه‌های جدیدی به‌صورت تصادفی ایجاد می‌شوند و سپس مقدار برازندگی این لانه‌ها محاسبه می‌شود. در ادامه یک لانه از بین لانه‌های موجود به‌صورت تصادفی انتخاب می‌شود و در صورتی که مقدار برازندگی لانه جدید بهتر باشد، لانه جدید جایگزین لانه موجود می‌شود. سپس هر پرنده در شعاع مکانی که می‌خواهد تخم‌گذاری کند، به دنبال لانه جدید یا همان جواب جدید می‌گردد. در این مرحله، الگوریتم جستجوی فاخته به پایان می‌رسد و عملکرد الگوریتم بهینه‌سازی ازدحام ذرات آغاز می‌شود. در این بخش، با استفاده از مراحل الگوریتم بهینه‌سازی ازدحام ذرات لانه‌های (داده‌های) جدیدی انتخاب می‌شوند. اگر لانه‌های جدید از لانه‌های موجود بهتر باشند، آنگاه لانه‌های جدید جایگزین می‌شوند. در واقع این عملیات تا زمانی که تعداد تکرار کافی باشد ادامه پیدا می‌کند. در انتها، از نتایج به‌دست‌آمده توسط هر دو الگوریتم به‌عنوان پارامترهای ورودی مدل پسا معماری کوکومو۲ استفاده می‌شود تا تخمین تلاش صورت گیرد.

این شبه کد حاوی تعدادی پارامتر است که در ادامه هر کدام از آن‌ها تعریف می‌شوند.

اولین پارامتر $f(x)$ است، که مشخص کننده تابع برازندگی می‌باشد. این تابع میزان خوب و یا بد بودن یک موقعیت را ارزیابی می‌کند. در واقع اهمیت هر زیستگاه با استفاده از این تابع تعیین می‌شود. t بیانگر تعداد زیستگاه‌ها می‌باشد، P_a حاوی حداکثر میزان شعاعی است که هر فاخته از زیستگاه خود می‌تواند تخم‌گذاری کند و r بیانگر تعداد تکرار روش پیشنهادی می‌باشد.

در ادامه، فرمولی وجود دارد که سرعت و موقعیت هر ذره از طریق آن بدست می‌آید. این فرمول حاوی پارامترهای زیر است:

در این فرمول $v_{i,d}$ بیانگر سرعت ذرات می‌باشد، w ضریب وزنی اینرسی[۲۷] است که بطور معمول برای آن مقداری در بازه‌ی [۰/۹- ۰/۴] درنظر گرفته می‌شود، $x_{i,d}$ موقعیت فعلی هر ذره را نشان می‌دهد، $pb_{i,d}$ بیانگر بهترین موقعیتی است که هر ذره تاکنون تجربه کرده و gb_d بهترین موقعیتی که تاکنون در بین همه‌ی ذرات تجربه شده است را نشان می‌دهد. به C_1 و C_2 ضرایب یادگیری گفته

می‌شود که مقدار آن‌ها در بازه‌ی [۲-۰] قرار دارد و $R(0,1)$ اعدادی تصادفی با توزیع یکنواخت[۲۸] در بازه‌ی [۱-۰] تولید می‌کند.

در انتهای این شبه کد فرمولی وجود دارد که از آن بمنظور تخمین تلاش استفاده می‌شود. این فرمول حاوی پارامترهای زیر است:

در این فرمول، A یک ضریب ثابت است که مقدار آن بصورت پیش‌فرض برابر با ۲/۹۴ می‌باشد و توان B حاصل جمع پنج فاکتور مقیاس است که اقتصادی و یا غیر اقتصادی بودن مقیاس نسبی پروژه‌های نرم‌افزاری در اندازه‌های مختلف را نشان می‌دهد. اگر B<1.0 باشد اقتصادی بودن مقیاس پروژه و اگر B>1.0 باشد غیر اقتصادی بودن مقیاس پروژه را نشان می‌دهد، همچنین اگر B=1.0 باشد مقیاس پروژه از لحاظ اقتصادی و یا غیر اقتصادی بودن در تعادل می‌باشد. در این فرمول Results حاوی نتایجی است که توسط الگوریتم‌های بهینه‌سازی ازدحام ذرات و جستجوی فاخته بدست آمده است. این نتایج عبارتند از: اندازه پروژه‌های نرم‌افزاری بر حسب تعداد خطوط کد و حاصل‌ضرب ۱۷ ضرب کننده تلاش.

۴- نتایج تجربی و ارزیابی کارآیی

در این بخش، مجموعه داده مورد استفاده، معیارهای ارزیابی، تنظیمات شبیه‌سازی و نتایج ارزیابی کارآیی عملکرد روش پیشنهادی ارائه شده است.

۴-۱- مجموعه داده

این مقاله از دو مجموعه داده کوکومو۸۱ و کوکوموناسا به‌منظور تخمین تلاش استفاده کرده است. ویژگی‌های این دو مجموعه داده در جدول ۱ بیان شده است. مجموعه داده کوکومو۸۱، حاوی ۶۳ پروژه نرم‌افزاری در زمینه‌های مختلف تجاری، علمی، سیستمی، بلادرنگ و پروژه‌های حمایتی می‌باشد [۱۵]. این مجموعه داده شامل ۱۷ ویژگی نرم‌افزاری است. ۱۵ ویژگی، ضرایب تلاش و یک ویژگی، اندازه نرم‌افزار برحسب هزاران خط کد به‌عنوان ویژگی‌های مستقل در این پژوهش می‌باشد. همچنین یک ویژگی، تلاش واقعی برای توسعه است که به‌عنوان ویژگی وابسته در نظر گرفته می‌شود [۵]. از ۱۷ ویژگی این مجموعه داده، ۱۵ ویژگی آن در مقیاسی متشکل از شش سطح که عبارت‌اند از: خیلی کم، کم، نرمال، زیاد، خیلی زیاد و فوق‌العاده زیاد اندازه‌گیری می‌شوند. بیش از سه دهه است که از این مجموعه داده به‌منظور ارزیابی تکنیک‌های تخمین تلاش استفاده می‌شود. مجموعه داده کوکوموناسا شامل ۶۰ پروژه از نرم‌افزارهای تولیدی مراکز مختلف سازمان ناسا است [۳۱]. این پروژه‌ها مربوط به دهه‌های ۱۹۸۰ و ۱۹۹۰ میلادی می‌باشند. مجموعه داده کوکوموناسا دقیقاً شامل همان ۱۷ ویژگی مجموعه داده کوکومو۸۱ است.

جدول ۱: ویژگی‌های مجموعه داده‌های کوکومو۸۱ و کوکوموناسا

شماره	نام ویژگی	نام کامل انگلیسی	معادل فارسی
۱	RELY	Required software reliability	قابلیت اطمینان موردنیاز برای نرم‌افزار
۲	DATA	Database size	اندازه بانک اطلاعاتی
۳	CPLX	Product complexity	پیچیدگی محصول
۴	TIME	Execution time constraint	محدودیت زمان اجرایی
۵	STOR	Main storage constraint	محدودیت حافظه اصلی
۶	VIRT	Virtual machine volatility	نوسانات ماشین مجازی
۷	TURN	Computer turnaround time	مدت زمان بین شروع کار نرم‌افزار تا دریافت خروجی
۸	ACAP	Analysts capability	توانایی تحلیل‌گرها
۹	AEXP	Application experience	تجربه‌ی کاربردی
۱۰	PCAP	Programmers capability	توانایی برنامه‌نویس‌ها
۱۱	VEXP	Virtual machine experience	تجربه‌ی ماشین مجازی
۱۲	LEXP	Programming language experience	تجربه‌ی زبان برنامه‌نویسی
۱۳	MODP	Modern programming practices	شیوه‌های نوین برنامه‌نویسی
۱۴	TOOL	Use of software tools	استفاده از ابزارهای نرم‌افزاری
۱۵	SCED	Required development schedule	زمان موردنیاز برای توسعه
۱۶	KLOC	Thousands(Kilo) lines of code	اندازه نرم‌افزار برحسب هزاران خط کد
۱۷	EFFORT	Actual effort	میزان تلاش واقعی برای توسعه نرم‌افزار

۴-۲- معیارهای ارزیابی کارآیی

در این مقاله، برای مقایسه کارآیی روش پیشنهادی با الگوریتم‌های پیشین از معیارهای ارزیابی زیر استفاده شده است:

۴-۲-۱ خطای نسبی[۲۹]

از جمله معیارهای ارزیابی مورد استفاده در مدل‌های مختلف می‌باشد. از این معیار به‌عنوان پایه و اساس محاسبه‌ی معیارهایی همچون: شدت خطای نسبی[۳۰] و متوسط شدت خطای نسبی در بسیاری از پژوهش‌های پیشین استفاده شده است. خطای نسبی مطابق رابطه (۱) تعریف شده است [۱۳].

$$RE = |Estimated - Actual| \quad (۱)$$

۴-۲-۲ شدت خطای نسبی

از این معیار در برخی از پژوهش‌ها به‌منظور ارزیابی دقت الگوریتم پیشنهادی بر روی نمونه پروژه‌های انتخابی استفاده می‌شود. به این صورت که میزان اختلاف بین میزان تلاش تخمین زده شده با میزان تلاش واقعی را تعیین می‌کند. شدت خطای نسبی از طریق رابطه (۲) محاسبه می‌شود.

$$MRE = \frac{|Estimated\ Effort - Actual\ Effort|}{Actual\ Effort} \quad (۲)$$

در رابطه (۲)، Estimated Effort، میزان تلاش تخمین زده شده توسط الگوریتم مورد ارزیابی و Actual Effort، میزان تلاش واقعی نمونه پروژه‌ی موردنظر در مجموعه داده می‌باشد.

۴-۲-۳ متوسط شدت خطای نسبی

از این معیار در برخی پژوهش‌های مربوط به حوزه تخمین تلاش، به‌منظور ارزیابی دقت الگوریتم پیشنهادی استفاده می‌شود. بدین منظور برای تمام نمونه پروژه‌های موردنظر، میانگین اختلاف تلاش تخمین زده شده توسط الگوریتم موردنظر با تلاش واقعی را مورد ارزیابی قرار می‌دهد. متوسط شدت خطای نسبی از طریق رابطه (۳) محاسبه می‌شود [۳۲].

$$\mathrm{MMRE} = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}} \frac{|\mathrm{Estimated(i)} - \mathrm{Actual(i)}|}{\mathrm{Actual(i)}} \quad (۳)$$

در رابطه (۳)، N تعداد پروژه‌های مورد ارزیابی، Actual میزان تلاش واقعی و Estimated میزان تلاش تخمین زده شده توسط الگوریتم مورد ارزیابی می‌باشد.

۴-۲-۴ درصد پیش‌بینی

این معیار احتمال اینکه میزان خطای تخمین زده شده، برای تمام نمونه‌های مورد ارزیابی کمتر و یا مساوی با مقدار X باشد را مشخص می‌نماید. درصد پیش‌بینی از طریق رابطه (۴) محاسبه می‌شود [۱۳, ۳۲].

$$\mathrm{PRED(X)} = \frac{\mathrm{M}}{\mathrm{N}} \quad (۴)$$

در رابطه (۴)، X بیانگر میزان اختلاف موردنظر است، به‌نحوی که در اغلب کارهای پژوهشی مقدار آن برابر با ۲۵٪ است. در این پژوهش برای X سه مقدار ۲۵٪، ۳۰٪ و ۴۰٪ در نظر گرفته شده است. در این رابطه، M بیانگر تعداد نمونه‌هایی است که اختلاف مقدار تلاش تخمین زده شده توسط الگوریتم مورد ارزیابی برای آن‌ها، با مقدار تلاش واقعی کوچک‌تر یا مساوی با مقدار X است. همچنین N بیانگر تعداد کل نمونه‌های مورد ارزیابی است. هر چه میزان درصد پیش‌بینی بیشتر شود، طبعاً میزان خطای الگوریتم مورد ارزیابی هم کمتر خواهد بود.

۴-۳- تنظیمات پارامترها و تحلیل حساسیت آن‌ها

یکی از مواردی که بر عملکرد الگوریتم‌های تکاملی تأثیرگذار است، تعیین بهینه پارامترهای الگوریتم‌ها می‌باشد. انتخاب نادرست پارامترها می‌تواند عملکرد الگوریتم‌ها را تحت تأثیر قرار داده و کارآیی الگوریتم‌ها را تا اندازه بسیار زیادی کاهش دهد. هدف این بخش از مقاله، انتخاب بهینه پارامترهای الگوریتم‌های بهینه‌سازی ازدحام ذرات و جستجوی فاخته می‌باشد. به همین منظور، پارامترهای روش پیشنهادی مورد بررسی قرار می‌گیرند.

اولین پارامتری که مورد بررسی قرار می‌گیرد، پارامتر تعداد تکرار است. این پارامتر در هر دو الگوریتم وجود دارد. هر چه تعداد تکرارها بیشتر باشد زمان اختصاص داده شده به الگوریتم‌ها برای همگرایی و در نتیجه شانس همگرایی الگوریتم‌ها افزایش پیدا می‌کند. البته باید توجه داشت که اگر تعداد تکرارها از یک حدی بیشتر شود، آنگاه پارامتر موردنظر تأثیری در همگرایی الگوریتم‌ها ندارد و روند همگرایی ثابت می‌ماند. این پارامتر در جدول ۲ مورد بررسی قرارگرفته است.

جدول ۲: بررسی پارامتر تعداد تکرار در روش پیشنهادی

تعداد تکرار	خطای پیش‌بینی
۱۰	۰/۰۳
۲۰	۰/۰۲
۵۰	۰/۰۰۴
۱۰۰	۰/۰۰۷
۲۰۰	۰/۰۰۷
۵۰۰	۰/۰۰۷

نتایج به‌دست‌آمده در جدول ۲ نشان می‌دهد که الگوریتم‌ها پس از ۱۰۰ تکرار تقریباً بهینه شده‌اند و افزایش تعداد تکرارها بیشتر از ۱۰۰، تأثیری بر قابلیت الگوریتم‌ها ندارد. پارامتر دیگری که در هر

دو الگوریتم وجود دارد پارامتر تعداد راه‌حل‌ها می‌باشد. این پارامتر در الگوریتم جستجوی فاخته با نام تعداد فاخته و در الگوریتم بهینه‌سازی ازدحام ذرات با نام تعداد ذرات شناخته می‌شود. واضح است که هر چقدر تعداد ذرات و فاخته‌ها بیشتر باشد شانس رسیدن به جواب بهینه افزایش پیدا می‌کند. برای مثال در الگوریتم جستجوی فاخته، اگر تعداد فاخته‌ها از یک حد معقول بالاتر رود شانس رسیدن به همگرایی کاهش می‌یابد. در جدول ۳، نتیجه بررسی این پارامتر در الگوریتم جستجوی فاخته نشان داده شده است.

جدول ۳: بررسی پارامتر تعداد فاخته‌ها در روش پیشنهادی

تعداد فاخته‌ها	خطای پیش‌بینی
۱۰	۰/۰۲۷
۲۰	۰/۰۲۱
۵۰	۰/۰۱۲
۱۰۰	۰/۰۰۸
۲۰۰	۰/۰۰۹
۵۰۰	۰/۰۱۸

نتایج به‌دست آمده در جدول ۳ نشان می‌دهد که هنگام تغییر تعداد فاخته از ۱۰ به ۱۰۰، خطا کاهش پیدا می‌کند اما هنگامی‌که این تعداد از ۱۰۰ به ۵۰۰ افزایش پیدا می‌کند میزان خطا بیشتر می‌شود. مقادیر پارامترهای الگوریتم‌های جستجوی فاخته و بهینه-سازی ازدحام ذرات در جدول‌های ۴ و ۵ نشان داده شده است.

جدول ۴: پارامترهای الگوریتم جستجوی فاخته

پارامتر	مقدار
تعداد فاخته‌ها	۱۰۰
تعداد کل لانه‌های موجود	۲۵
نرخ اکتشاف	۰/۲۵
ضریب حرکت	۲
تعداد تکرار	۱۰۰

مقادیر پارامترهای تعداد کل لانه‌های موجود، نرخ اکتشاف و ضریب حرکت در الگوریتم جستجوی فاخته بر اساس مقاله پایه [۲] انتخاب شده‌اند و مقادیر پارامترهای تعداد فاخته و تعداد تکرار به‌صورت بهینه انتخاب شده است.

جدول ۵: پارامترهای الگوریتم بهینه‌سازی ازدحام ذرات

پارامتر	مقدار
تعداد ذرات	۱۰۰
ضریب وزنی اینرسی	[۰/۴ -۰/۹]
ضرایب c_1 و c_2	۲
ضرایب r_1 و r_2	[۰-۱]
تعداد تکرار	۱۰۰

- ضریب وزنی اینرسی: برای این پارامتر مقداری در بازه‌ی [۰/۴ -۰/۹] در نظر گرفته می‌شود [۳۳]. مقدار این پارامتر در تکرارهای مختلف ثابت نمی‌باشد، بلکه در تکرارهای ابتدایی مقدار آن بزرگ و در تکرارهای آتی مقدار آن به‌تدریج کم می‌گردد. این موضوع باعث می‌شود که کل فضای مسئله با دقت بیشتری مورد جستجو قرار گیرد.
- ضرایب c_1 و c_2: به این ضرایب، ضرایب یادگیری گفته می‌شود که مقدار آن‌ها در بازه‌ی [۲-۰] قرار دارد [۳۳]. ضریب c_1 مشخص می‌کند که بهترین موقعیت تجربه شده توسط هر ذره به چه میزان در یافتن تابع هدف اثرگذار می‌باشد و همچنین ضریب c_2 مشخص می‌کند که بهترین موقعیت تجربه شده توسط سراسر ذرات به چه میزان در یافتن تابع هدف تأثیرگذار می‌باشد، لذا در این پژوهش برای هر دو ضریب مقدار ۲ در نظر گرفته شده تا با دقت بیشتری جواب بهینه نهایی به‌دست آید.
- ضرایب r_1 و r_2: این دو ضریب با توزیع یکنواخت، اعدادی تصادفی در بازه‌ی [۱-۰] تولید می‌کنند [۳۳]. مقادیر پارامترهای تعداد ذرات و تعداد تکرار به‌صورت بهینه انتخاب شده‌اند.

در این مقاله از مدل پسا معماری کوکومو۲ برای تخمین تلاش استفاده شده است. در واقع مدل پسا معماری کوکومو۲، از ویژگی‌های نرم‌افزاری که توسط روش پیشنهادی انتخاب شده به‌عنوان پارامترهای ورودی خود، جهت تخمین تلاش استفاده می‌کند. این مدل به‌منظور تخمین تلاش از سه پارامتر بسیار مهم: اندازه نرم‌افزار، فاکتورهای مقیاس[۳۱] و محرک‌های هزینه[۳۲] استفاده می‌کند.

- ✓ اندازه نرم‌افزار: در این مدل اندازه نرم‌افزار بر اساس مقیاس تعداد خطوط کد به‌دست می‌آید. اندازه هرکدام از پروژه‌های نرم‌افزاری در مجموعه داده‌های کوکومو۸۱ و کوکوموناسا مشخص شده است. این پارامتر جزئی از ویژگی‌های نرم‌افزاری است که توسط روش پیشنهادی انتخاب می‌شود.
- ✓ فاکتورهای مقیاس: فاکتورهای مقیاس تأثیر بسیار زیادی بر روی تخمین تلاش دارند. کوکومو۲ شامل ۵ فاکتور مقیاس: سابقه[۳۳]، انعطاف‌پذیری توسعه[۳۴]، تحلیل ریسک[۳۵]، انسجام تیمی[۳۶] و بلوغ فرآیند[۳۷] می‌باشد. به هرکدام از این فاکتورها بر اساس میزان تأثیری که بر روی تخمین تلاش دارند، در شش سطح از خیلی کم تا فوق‌العاده زیاد یک

وزن اختصاص می‌دهند. در این مقاله هر یک از فاکتورهای مقیاس همانند مقاله پایه [۲] به‌صورت جزئی بر روی تخمین تلاش تأثیر می‌گذارند، با توجه به این موضوع به هرکدام از فاکتورهای مقیاس یک مقدار اختصاص داده شده است. مقادیر هر کدام از فاکتورهای مقیاس توسط بَری بوهم[۳۸] [۳۴] ارائه شده که در جدول ۶ نشان داده می‌شود.

جدول ۶: مقادیر فاکتورهای مقیاس

فاکتورهای مقیاس	مقادیر
سابقه	۳/۷۲
انعطاف‌پذیری توسعه	۳/۰۴
تحلیل ریسک	۴/۲۴
انسجام تیمی	۳/۲۹
بلوغ فرآیند	۴/۶۸

- ✓ محرک‌های هزینه: هر کدام از محرک‌های هزینه یک ویژگی از پروژه نرم‌افزاری را نشان می‌دهند. در واقع محرک‌های هزینه شامل همان ویژگی‌های نرم‌افزاری هستند که توسط روش پیشنهادی انتخاب می‌شوند.

۴-۴- نتایج ارزیابی کارآیی

پس از تعیین پارامترها و مقادیر آن‌ها در بخش پیشین، در این بخش روش پیشنهادی مورد تحلیل و ارزیابی قرار می‌گیرد. به این منظور، میزان دقت روش پیشنهادی به‌صورت جداگانه بر روی مجموعه داده‌های کوکومو۸۱ و کوکوموناسا بر اساس معیارهای متوسط شدت خطای نسبی و درصد پیش‌بینی محاسبه می‌گردد.

مجموعه داده کوکومو۸۱ شامل اطلاعات ۶۳ پروژه نرم‌افزاری و مجموعه داده کوکوموناسا حاوی اطلاعات ۶۰ پروژه نرم‌افزاری می‌باشد. همان‌طور که در جداول ۷ و ۸ ارائه شده، نتایج معیارهای متوسط شدت خطای نسبی و درصد پیش‌بینی برای سه حالت ۲۵٪، ۳۰٪ و ۴۰٪ محاسبه شده است. با انجام یک مقایسه میان نتایج ارائه شده توسط روش پیشنهادی در هر دو مجموعه داده، می‌توان نتیجه گرفت که مقدار متوسط شدت خطای نسبی در مجموعه داده کوکوموناسا کم‌تر می‌باشد.

۵- مقایسه روش پیشنهادی با سایر روش‌ها

مهم‌ترین هدف این بخش از مقاله، ارزیابی کارآیی روش پیشنهادی و اثبات ادعای برتری عملکرد آن در مقایسه با سایر روش‌های مرتبط، با انجام یک مقایسه میان کارآیی روش پیشنهادی و روش ارائه‌شده در [۲] بر اساس معیارهای متوسط شدت خطای نسبی و درصد پیش‌بینی می‌باشد. نتایج این مقایسه در دو بخش به‌صورت جداگانه برای مجموعه داده‌های کوکومو۸۱ و کوکوموناسا ارائه شده است.

۵-۱- نتایج مقایسه روش‌ها بر روی مجموعه داده کوکومو۸۱

در این بخش روش پیشنهادی بر اساس معیارهای متوسط شدت خطای نسبی و درصد پیش‌بینی با روش ارائه شده در [۲] مقایسه شده، که نتایج بدست آمده در جدول ۷ گزارش شده است.

جدول ۷: مقایسه متوسط شدت خطای نسبی و میزان درصد پیش‌بینی روش پیشنهادی با روش ارائه شده در [۲] برای حالات متفاوت در مجموعه داده کوکومو۸۱

روش‌های تخمین تلاش	متوسط شدت خطای نسبی	درصد پیش‌بینی (٪۲۵)	درصد پیش‌بینی (٪۳۰)	درصد پیش‌بینی (٪۴۰)
روش پیشنهادی	۰/۳۹۲	٪۷۴/۵۳	٪۷۸/۱۸	٪۸۶/۸۹
روش ارائه شده در [۲]	۰/۵۶۹	٪۶۶/۶۶	٪۷۰/۱۴	٪۷۸/۲۳

نتایج مقایسه نشان می‌دهد که مقدار متوسط شدت خطای نسبی روش پیشنهادی برای مجموعه داده کوکومو۸۱ نسبت به روش ارائه شده در [۲]، به اندازه ۰/۱۷۷ کاهش یافته است.

روش ارائه شده در [۲] از همگرایی مناسبی برخوردار نمی‌باشد؛ بنابراین بهینه محلی به‌جای بهینه سراسری در خروجی مشخص می‌شود. لذا با ترکیب الگوریتم‌های بهینه‌سازی ازدحام ذرات و جستجوی فاخته، فضای مسئله دقیق‌تر مورد جستجو قرار می‌گیرد و امکان دستیابی به بهینه سراسری افزایش پیدا می‌کند. این موضوع باعث می‌گردد که تخمین تلاش به میزان تلاش واقعی نزدیک‌تر شود؛ بنابراین با نزدیک‌تر شدن تخمین تلاش به واقعیت، مقدار متوسط شدت خطای نسبی کاهش می‌یابد.

در ادامه میزان درصد پیش‌بینی روش پیشنهادی و روش ارائه شده در [۲]، بر اساس سه حالت ۲۵٪، ۳۰٪ و ۴۰٪ بر روی مجموعه داده کوکومو۸۱ مقایسه شده، که در جدول ۷ نشان داده شده است.

ارزیابی صورت گرفته نشان می‌دهد که میزان درصد پیش‌بینی روش پیشنهادی در مجموعه داده کوکومو۸۱ نسبت به روش ارائه شده در [۲]، در حالت ۲۵٪ به‌اندازه ۷/۸۷ درصد، در حالت ۳۰٪ به‌اندازه ۸/۰۴ درصد و در حالت ۴۰٪ به‌اندازه ۸/۶۶ درصد افزایش یافته است.

همان‌طور که نتایج مقایسه نشان می‌دهد، روش پیشنهادی تلاش می‌کند که هر دو الگوریتم به‌صورت موازی اجرا شوند تا در هر تکرار جواب‌های به‌دست آمده توسط الگوریتم جستجوی فاخته با استفاده

از الگوریتم بهینه‌سازی ازدحام ذرات بیشتر مورد واکاوی قرار گیرد. در این حالت جواب‌های به‌دست آمده در هر دو الگوریتم بین هم به اشتراک گذاشته می‌شوند تا فضای مسأله دقیق‌تر مورد جستجو قرار گیرد. این موضوع باعث افزایش میزان درصد پیش‌بینی می‌شود.

۵-۲- نتایج مقایسه روش‌ها بر روی مجموعه داده کوکوموناسا

در این بخش به‌منظور ارزیابی کارآیی روش پیشنهادی، معیارهای متوسط شدت خطای نسبی و درصد پیش‌بینی با روش ارائه شده در [۲] مقایسه شده است. این مقایسه در جدول ۸ گزارش شده است.

جدول ۸: مقایسه متوسط شدت خطای نسبی و میزان درصد پیش‌بینی روش پیشنهادی با روش ارائه شده در [۲] برای حالات متفاوت در مجموعه داده کوکوموناسا

روش‌های تخمین تلاش	متوسط شدت خطای نسبی	درصد پیش‌بینی (۲۵٪)	درصد پیش‌بینی (۳۰٪)	درصد پیش‌بینی (۴۰٪)
روش پیشنهادی	۰/۱۹۷	٪۷۳/۸۷	٪۷۷/۱۲	٪۸۷/۵۸
روش ارائه شده در [۲]	۰/۳۴۸	٪۶۶/۳۲	٪۶۹/۱۴	٪۷۹/۴۷

ارزیابی نتایج ارائه شده در جدول ۸ نشان می‌دهد که متوسط شدت خطای نسبی روش پیشنهادی در مجموعه داده کوکوموناسا نسبت به روش ارائه شده در [۲]، به اندازه ۰/۱۵۱ کاهش یافته است. علت بهبود به‌دست آمده واضح می‌باشد، زیرا هنگام ترکیب الگوریتم‌های بهینه‌سازی ازدحام ذرات و جستجوی فاخته، فضای مسئله بهتر مورد جستجو قرار می‌گیرد و در نتیجه امکان دستیابی به بهینه سراسری افزایش پیدا می‌کند. این موضوع باعث می‌گردد که تشخیص ویژگی‌های نرم‌افزاری با دقت بیشتری صورت گیرد. با تشخیص صحیح ویژگی‌های نرم‌افزاری، میزان تخمین تلاش مورد نیاز برای توسعه پروژه‌های نرم‌افزاری به واقعیت نزدیک‌تر می‌شود. در این صورت مقدار متوسط شدت خطای نسبی کاهش می‌یابد.

در ادامه میزان درصد پیش‌بینی روش پیشنهادی با روش ارائه شده در [۲]، بر روی مجموعه داده کوکوموناسا بر اساس سه حالت ٪۲۵، ٪۳۰ و ٪۴۰ مقایسه شده، که در جدول ۸ ارائه شده است.

ارزیابی صورت گرفته در جدول ۸ نشان می‌دهد که میزان درصد پیش‌بینی روش پیشنهادی در مجموعه داده کوکوموناسا نسبت به روش ارائه شده در [۲]، در حالت ٪۲۵ به‌اندازه ۷/۵۵ درصد، در حالت ٪۳۰ به‌اندازه ۷/۹۸ درصد و در حالت ٪۴۰ به‌اندازه ۸/۱۱ درصد افزایش یافته است؛ بنابراین نتایج مقایسه نشان می‌دهد که روش پیشنهادی با مبادله‌ی داده‌ها بین هر دو الگوریتم، فضای مسئله را دقیق‌تر مورد جستجو قرار می‌دهد و امکان دستیابی به نتایج بهینه افزایش پیدا می‌کند.

به‌طور کلی نتایج ارزیابی انجام شده نشان می‌دهد که روش پیشنهادی نسبت به روش ارائه شده در [۲]، هنگام عمل تخمین تلاش توسعه نرم‌افزار بر روی مجموعه داده‌های کوکومو۸۱ و کوکوموناسا عملکرد بهتری داشته و از دقت بالاتری برخوردار می‌باشد. در واقع در هر دو مجموعه داده، مقدار متوسط شدت خطای نسبی توسط روش پیشنهادی در مقایسه با روش ارائه شده در [۲]، کاهش یافته و میزان درصد پیش‌بینی افزایش یافته است.

۶- نتیجه‌گیری و کارهای آتی

در این مقاله یک روش ترکیبی کارآمد برای بهبود تخمین تلاش توسعه نرم‌افزار پیشنهاد گردید. روش پیشنهادی، الگوریتم‌های بهینه‌سازی ازدحام ذرات و جستجوی فاخته را ترکیب می‌کند. هدف از پیشنهاد ترکیب این الگوریتم‌ها، اجرای متوالی هر دو الگوریتم می‌باشد به گونه‌ای که در هر تکرار پاسخ به‌دست آمده توسط الگوریتم جستجوی فاخته با استفاده از الگوریتم بهینه‌سازی ازدحام ذرات، بیشتر مورد واکاوی قرار گیرد. الگوریتم بهینه‌سازی ازدحام ذرات به‌منظور کاهش متوسط شدت خطای نسبی در این ترکیب استفاده می‌شود. به این ترتیب می‌توان پارامترهای مدل پسا معماری کوکومو۲ را بهینه نموده و دقت انتخاب ویژگی‌های نرم‌افزاری را بهبود داد. همچنین استفاده از ترکیب پیشنهادی به دلیل اشتراک‌گذاری جواب‌های بهینه بین الگوریتم‌ها، منجر به افزایش میزان درصد پیش‌بینی می‌شود. نتایج ارزیابی کارآیی روش پیشنهادی در مقایسه با سایر روش‌های پیشین نشان می‌دهد که روش پیشنهادی در زمینه‌ی تخمین تلاش توسعه نرم‌افزار از دقت بالاتری برخوردار می‌باشد و عملکرد بهتری ارائه می‌کند.

ما قصد داریم مسائل زیر را به‌عنوان کارهای آتی در پژوهش خود مورد بررسی قرار دهیم:

- ارائه الگوریتم‌های فراابتکاری یا ابتکاری جدید مانند الگوریتم دیفرانسیل[۳۹] و گرانش.
- کاربرد خوشه‌بندی[۴۰] برای کاهش ابعاد مسأله، کاهش خطا و بهبود کارآیی.
- استفاده از نظریه فازی[۴۱] برای بهبود فرآیند بهینه‌سازی و کاهش زمان تخمین تلاش

مراجع

[1] J. Popović and D. Bojić, "A comparative evaluation of effort estimation methods in the software life cycle," Computer Science and Information Systems, vol. 9, no. 1, pp. 455-484, 2012.

[2] S. Kumari and S. Pushkar, "Cuckoo search based hybrid models for improving the accuracy of software effort estimation," Microsystem Technologies, vol. 24, no. 12, pp. 4767-4774, 2018.

[3] M. Padmaja and D. Haritha, "Software Effort Estimation using Meta Heuristic Algorithm," International Journal of Advanced Research in Computer Science, vol. 8, no. 5, 2017.

[4] P. Singal, A. C. Kumari, and P. Sharma, "Estimation of software development effort: A Differential Evolution Approach," Procedia Computer Science, vol. 167, pp. 2643-2652, 2020.

[5] P. S. Kumar and H. Behera" ,Estimating Software Effort Using Neural Network: An Experimental Investigation," in Computational Intelligence in Pattern Recognition: Springer, 2020, pp. 165-180.

[6] A. Sinhal and B. Verma, "Software Development Effort Estimation: A Review," International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), vol. 3, no. 6, pp. 1120-1135, 2013.

[7] M. Jørgensen, "What we do and don't know about software development effort estimation," IEEE Software, vol. 31, no. 2, pp . ۲۰۱۴ ,۳۷-۴۰.

[8] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, "An effective approach for software project effort and duration estimation with machine learning algorithms," Journal of Systems and Software, vol. 137, pp. 184-196, 2018.

[9] M. Usman, R. Britto, L.-O. Damm, and J. Börstler, "Effort estimation in large-scale software development: An industrial case study," Information and Software technology, vol. 99, pp. 21-40, 2018.

[10] M. Aljohani and M. Qureshi, "Comparative study of software estimation techniques," International Journal of Software Engineering & Applications (IJSEA), vol. 8, no. 6, 2017.

[11] A. K. Bardsiri and S. M. Hashemi, "Software effort estimation: a survey of well-known approaches," International Journal of Computer Science Engineering (IJCSE), vol. 3, no. 1, pp. 46-50, 2014.

[12] F. A. Amazal and A. Idri, "Estimating software development effort using fuzzy clustering-based analogy," Journal of Software: Evolution and Process, p. e2324, 2020.

[13] T. T. Khuat and M. H .Le, "Optimizing parameters of software effort estimation models using directed artificial bee colony algorithm," Informatica, vol. 40, no. 4, 2016.

[14] H. R. Maier, S. Razavi, Z. Kapelan, L. S. Matott, J. Kasprzyk, and B. A. Tolson, "Introductory overview: Optimization using evolutionary algorithms and other metaheuristics," Environmental modelling & software, vol. 114, pp. 195-213, 2019.

[15] V. Venkataiah, R. Mohanty, J. Pahariya, and M. Nagaratna, "Application of ant colony optimization techniques to predict software cost estimation," in Computer Communication, Networking and Internet Security: Springer, 2017, pp. 315-325.

[16] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in 2009 World congress on nature & biologically inspired computing (NaBIC), 2009, pp. 210-214: Ieee.

[17] X.-S. Yang, "Chapter 2 - Analysis of Algorithms," in Nature-Inspired Optimization Algorithms, X.-S. Yang, Ed. Oxford: Elsevier, 2014, pp. 23-44.

[18] M. Guerrero-Luis, F. Valdez, and O. Castillo, "A Review on the Cuckoo Search Algorithm," in Fuzzy Logic Hybrid Extensions of Neural and Optimization Algorithms: Theory and Applications, O. Castillo and P. Melin, Eds. Cham: Springer International Publishing, 2021, pp. 113-124.

[19] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: an overview," Soft Computing, vol. 22, no. 2, pp. 387-408, 2018.

[20] J.-C. Lin and H.-Y. Tzeng, "Applying particle swarm optimization to estimate software effort by multiple factors software project clustering," in 2010 International Computer Symposium (ICS2010), 2010, pp. 1039-1044: IEEE.

[21] M. Kaur and S. K. Sehra, "Particle swarm optimization based effort estimation using Function Point analysis," in 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014, pp. 140-145: IEEE.

[22] J. Murillo-Morera, C. Quesada-López, C. Castro-Herrera, and M. Jenkins, "A genetic algorithm based framework for software effort prediction," Journal of software engineering research and development, vol. 5, no. 1, pp. 1-33, 2017.

[23] D. Wu, J. Li, and C. Bao, "Case-based reasoning with optimized weight derived by particle swarm optimization for software effort estimation," Soft Computing, vol. 22, no. 16, pp. 5299-5310, 2018.

[24] S. S. Gautam and V. Singh, "The state-of-the-art in software development effort estimation," Journal of Software: Evolution and Process, vol. 30, no. 12, p. e1983, 2018.

[25] T. Menzies, Y. Yang, G. Mathew, B. Boehm, and J. Hihn, "Negative results for software effort estimation," Empirical Software Engineering, vol. 22, no. 5, pp. 2658-2683, 2017.

[26] V. Beiranvand, W. Hare, and Y. Lucet, "Best practices for comparing optimization algorithms," Optimization and Engineering, vol. 18, no. 4, pp. 815-848, 2017.

[27] M. A .Shah, D. N. A. Jawawi, M. A. Isa, M. Younas, A. Abdelmaboud, and F. Sholichin, "Ensembling artificial bee colony with analogy-based estimation to improve software development effort prediction," IEEE Access, vol. 8, pp. 58402-58415, 2020.

[28] H. Mustapha and N. Abdelwahed, "Investigating the use of random forest in software effort estimation," Procedia computer science, vol. 148, pp. 343-352, 2019.

[29] S. Ezghari and A. Zahi, "Uncertainty management in software effort estimation using a consistent fuzzy analogy-based method," Applied Soft Computing, vol. 67, pp. 540-557, 2018.

[30] F. Zare, H. K. Zare, and M. S. Fallahnezhad, "Software effort estimation based on the optimal Bayesian belief network," Applied Soft Computing, vol. 49, pp. 968-980, 2016.

[31] M. Shanker, J. Jaya, and K. Thanushkodi, "An Effective Approach to Software Cost Estimation Based on Soft Computing Techniques," International Arab Journal of Information Technology (IAJIT), vol. 12, 2015.

[32] B. V. KHATIBI and M. Dorosti, "An Improved COCOMO based Model to Estimate the Effort of Software Projects," 2016.

[33] K.-L. Du and M. Swamy, "Particle Swarm Optimization," in Search and Optimization by Metaheuristics: Springer, 2016, pp. 153-173.

[34] A. Trendowicz and R. Jeffery, "Software project effort estimation," Foundations and Best Practice Guidelines for Success, Constructive Cost Model–COCOMO pags, vol. 12, pp. 277-293, 2014.

پاورقی‌ها:

[1] Software Development Effort Estimation (SDEE)
[2] Software Development Life Cycle (SDLC)
[3] Mean Magnitude of Relative Error (MMRE)
[4] Percentage of Prediction (PRED)
[5] Cuckoo Search (CS)
[6] Artificial Neural Network (ANN)
[7] Objective function
[8] Genetic algorithm (GA)
[9] Lévy Flights
[10] Global optimum
[11] Local optimum
[12] Particle Swarm Optimization (PSO)
[13] Ant colony optimization (ACO)
[14] http://promise.site.uottawa.ca/SERepository/datasets/cocomo81.arff
[15] http://promise.site.uottawa.ca/SERepository/datasets/cocomonas.arff
[16] Meta-heuristic
[17] Bat Algorithm (BA)
[18] Kemerer
[19] Grey Relational Analysis (GRA)
[20] Artificial Bee Colony (ABC)
[21] Premature convergence
[22] Random Forest (RF)
[23] Regression Tree (RT)
[24] Fuzzy analogy
[25] Post-Architecture model
[26] Fitness function
[27] Inertia weight
[28] Uniform distribution
[29] Relative Error (RE)
[30] Magnitude Relative Error (MRE)
[31] Scale Factors (SF)
[32] Cost Drivers
[33] Precedentedness (PREC)
[34] Development Flexibility (FLEX)
[35] Risk Resolution (RESL)
[36] Team Cohesion (TEAM)
[37] Process Maturity (PMAT)
[38] Barry Boehm
[39] Differential algorithm
[40] Clustering
[41] Fuzzy theory