DOR :20.1001.1.2538161.1401.7.2.6.3

دریافت مقاله: ۱۴۰۰/۱۲/۲۰
پذیرش مقاله:۱۴۰۱/۰۲/۲۱

نوع مقاله: پژوهشی

بهبود تخمین هزینه توسعه نرم‌افزار مبتنی بر تشابه با استفاده از مدل تکاملی یادگیر

تقی جاودانی گندمانی*
گروه علوم کامپیوتر، دانشگاه شهرکرد، شهرکرد، ایران
پست الکترونیکی: javdani@sku.ac.ir

مائده دشتی
هسته پژوهشی علوم داده، گروه علوم کامپیوتر، دانشگاه شهرکرد، شهرکرد، ایران
پست الکترونیکی: maedeh.dashti@sku.ac.ir

چکیده

یکی از مهم‌ترین و بحرانی‌ترین عوامل در توسعه پروژه‌های نرم‌افزاری تخمین مناسب هزینه‌ها است. این فعالیت که پیش از آغاز پروژه و در مراحل اولیه باید انجام گیرد به دلیل درگیر بودن عوامل متعدد انسانی، فنی و سازمانی همواره با چالش‌ها و مشکلات متعددی روبرو است. دیدگاه‌ها و روش‌های متعددی در خصوص نحوه انجام تخمین ارائه شده است که یکی از مهم‌ترین آن‌ها دیدگاه مبتنی بر تشابه می‌باشد. در این دیدگاه از روش‌های متفاوتی از جمله بهره‌گیری از ویژگی‌های مناسب و وزن‌دهی ویژگی‌ها در جهت افزایش دقت تخمین استفاده می‌گردد. این تحقیق برای بهبود تخمین هزینه توسعه نرم‌افزار، میزان تاثیر الگوریتم تکاملی یادگیر[1] بر بهینه‌سازی وزن ویژگی‌ها را مورد بررسی قرار داده است و اقدام به ارائه راهکاری نوین در این خصوص نموده است. در این تحقیق میزان اثربخشی الگوریتم بر روی دو مجموعه داده Desharnais و Maxwell بررسی شده است و از معیارهای MMRE، PRED(۰,۲۵) و MdMRE برای ارزیابی و مقایسه روش پیشنهادی با دیگر الگوریتم‌های تکاملی استفاده شده است. نتایج نشان می‌دهد این الگوریتم توانسته است بهبود قابل توجهی را به‌دست آورد.

واژه‌های کلیدی: تخمین هزینه نرم‌افزار، مدل تکاملی یادگیر، تخمین مبتنی بر تشابه، بهینه‌سازی وزن ویژگی‌ها

۱- مقدمه

تخمین مناسب هزینه‌های توسعه محصولات نرم‌افزاری همواره یکی از دغدغه‌های جدّی مدیران شرکت‌ها و پروژه‌های نرم‌افزاری بوده است. به همین دلیل همواره تلاش‌های قابل توجهی در جهت پیش‌بینی دقیق‌تر هزینه‌ها برای زمان‌بندی و برنامه‌ریزی موثر و مدیریت کیفیت محصول انجام شده است. همچنین روش‌ها و مدل‌هایی برای ساده‌سازی عملیات تخمین هزینه‌ها ارائه شده است. اما امروزه به دلیل رشد چشمگیر تعداد پروژه‌های نرم‌افزاری و فضای رقابتی جدّی‌تر در این

* نویسندهٔ مسئول

1- Learnable Evolution Model

صنعت و از طرف دیگر متغیر بودن هزینه‌ها بخصوص در روشگان‌های چابک و وجود عوامل اثرگذار بیشتر در توسعه پروژه‌های نرم‌افزاری، دستیابی به تخمین دقیق‌تر هنوز به شکل جدّی‌تری مورد توجه متخصصین نرم‌افزار می‌باشد [۱]. در واقع، مدیران شرکت‌ها و پروژه‌های نرم‌افزاری نیاز به یک سری روش‌های قابل اطمینان برای تخمین مناسب دارند تا بتوانند تصمیم‌گیری بهتری برای مدیریت چرخه حیات محصولات نرم‌افزاری و ارائه آن‌ها به بازار داشته باشند.

مرور ادبیات نشان می‌دهد که محققان بر مبنای اصول و عوامل مختلف، دسته‌بندی‌های متفاوتی از روش‌ها و دیدگاه‌های تخمین اندازه نرم‌افزار ارائه نموده‌اند. یورگنسن و شپرد[۲] در طی یک بررسی سیستماتیک بین ۳۰٤ مقاله از ۷٦ مجله، ۱۱ روش تخمین متفاوت را شناسایی نموده و آن‌ها را به دو دسته مدل‌های پارامتری و غیرپارامتری تقسیم‌بندی نموده‌اند [۲]. در مدل‌های پارامتری تخمین براساس تجزیه و تحلیل آماری و یا عددی از مجموعه داده‌های تاریخی است. در مدل‌های غیرپارامتری تخمین براساس اصول بهینه‌سازی و روش‌های هوش مصنوعی مانند شبکه‌های عصبی مصنوعی، الگوریتم ژنتیک، تخمین مبتنی بر تشابه یا مبتنی بر مورد، درخت تصمیم و مواردی از این دست می‌باشد.

در بین روش‌ها و دیدگاه‌های مختلف تخمین اندازه نرم‌افزار، روش تخمین مبتنی بر تشابه یکی از محبوب‌ترین روش‌ها می‌باشد. این روش که در واقع یک مدل مبتنی بر مورد است، اولین بار توسط استنبرگ[۳] معرفی شد و سپس از آن برای بهبود تخمین هزینه‌ها در توسعه نرم‌افزار استفاده گردید. با توجه به جنبه‌های مختلفی که در این روش وجود دارد، استفاده از روش‌های یادگیری ماشین و محاسبات نرم برای بالا بردن دقت تخمین در این روش، در سال‌های اخیر مورد توجه محققین قرار گرفته است [۳, ٤]. این روش‌ها می‌توانند به‌صورت مستقیم از طریق فرایند انتخاب پروژه یا وزن‌دهی به ویژگی‌ها مورد استفاده قرار گیرند [٥] و یا به‌طور غیرمستقیم از طریق روش‌های یادگیری ماشین مانند شبکه‌های عصبی مصنوعی یا شبکه‌های عصبی فازی اعمال شوند [٦].

یکی از الگوریتم‌های نوظهور در هوش مصنوعی مدل تکاملی یادگیر است. این الگوریتم یکی از الگوریتم‌های محاسبات تکاملی غیرداروینی است که از یادگیری ماشین برای هدایت فرآیند تکاملی استفاده می‌کند و نوع جدیدی از اپراتورها را برای ایجاد جمعیت اولیه پیشنهاد می‌کند که به لحاظ تعداد مراحل تکامل، می‌تواند سرعت ۲ تا چند برابری را به‌دست آورد [۷]. به همین دلیل، به‌کارگیری این الگوریتم در کاربردهای مختلف مورد توجه محققان بوده و نتایج مناسبی را در این کاربردها در پی داشته است [۸, ۹]. به دلیل دستیابی به نتایج مناسب در حل مسائل بهینه‌سازی پیچیده در دنیای واقعی، در این مطالعه از این الگوریتم به‌عنوان پایه‌ای برای بهبود تخمین هزینه توسعه نرم‌افزار در روش مبتنی بر تشابه استفاده شده است. راهکار ارائه شده در این مطالعه با بهره‌گیری از الگوریتم تکاملی یادگیر، براساس معیارهای رایج در این حوزه مورد ارزیابی قرار گرفته و نتایج نشان‌دهنده افزایش دقت تخمین به‌دست آمده توسط این راهکار در مقایسه با سایر الگوریتم‌های تکاملی می‌باشند.

ادامه این مقاله در ۸ بخش ارائه می‌شود. در بخش ۲ به معرفی اجمالی روش تخمین مبتنی بر تشابه پرداخته می‌شود و در ادامه در بخش ۳ کارهای مرتبط انجام شده در این حوزه مورد بررسی قرار خواهند گرفت. در بخش ٤ الگوریتم مدل تکاملی یادگیر ارائه شده است. پس از آن در بخش ٥ درباره راهکار پیشنهادی تشریح می‌گردد. سپس معیارهای ارزیابی، مجموعه داده‌ها و نحوه پیاده‌سازی آن در بخش ٦ آمده است. در بخش ۷، نتایج و آزمایش‌های تجربی با استفاده از مدل پیشنهادی ارائه خواهد شد. در نهایت، در بخش ۸ نتیجه گیری ارائه خواهد شد.

2- Jorgensen and Shepperd
3- Sternberg

۲. روش تخمین مبتنی بر تشابه

روش تخمین مبتنی بر تشابه رایج‌ترین روش غیرالگوریتمی است. در این روش برای تخمین یک پروژه جدید باید آن را با پروژه‌های مشابهی که در گذشته انجام شده است، مقایسه نمود. تخمین مبتنی بر تشابه شامل چهار بخش دادگان تاریخی[4]، تابع شباهت[5]، تعداد نزدیک‌ترین همسایه[6]، و تابع پاسخ[7] می‌باشد و فرایند تخمین مبتنی بر تشابه شامل گام‌های زیر است:

۱. ایجاد دادگان تاریخی که از طریق دادگان واقعی یا مصنوعی در دسترس می‌باشد.

۲. به‌دست آوردن ویژگی‌های مربوط به پروژه‌ای جدید به‌طوری که با دادگان تاریخی مطابقت داشته باشد.

۳. استفاده از یک تابع شباهت از پیش تعریف شده مانند توابع فاصله اقلیدسی و منهتن تا بتوان پروژه‌های مشابه با پروژه جدید را بازیابی نمود.

٤. هزینه پروژه جدید با استفاده از تابع پاسخ تخمین زده می‌شود.

در زیر هر کدام از اجزای سیستم تخمین مبتنی بر تشابه به‌طور جداگانه ارائه شده است.

۲-۱- تابع شباهت

هسته اصلی روش تخمین مبتنی بر تشابه تابع شباهت است که در آن میزان تشابه بین دو پروژه مختلف محاسبه می‌شود. فرم کلی تابع شباهت به شکل معادله (۱) بیان می‌شود:

(۱)

$$sim(p.p') = f\left(Lsim(f_1.f_1').Lsim(f_2.f_2').\dots.Lsim(f_n.f_n')\right)$$

در این معادله p و p' نشان‌دهنده دو پروژه جدید و پروژه قدیمی در مجموعه داده است، f_i و f_i' نشان‌دهنده مقدار ویژگی iام در پروژه‌های ذکر شده می‌باشد، n نشان‌دهنده تعداد کل ویژگی‌ها در هر پروژه است و تابع $Lsim(.)$ میزان شباهت بین دو ویژگی متناظر از پروژه را محاسبه می‌نماید. توابع $Lsim(.)$ و $f(.)$ نشان‌دهنده ساختار عمومی تابع شباهت می‌باشند. شباهت دو پروژه با استفاده از فاصله اقلیدسی با استفاده از معادله (۲) به‌دست می‌آید.

$$sim(p.p') = \frac{1}{\left[\sqrt{\sum_{i=1}^{n} w_i Dis(f_i.f_i')+\delta}\right]} \quad \delta = 0.0001 \qquad (۲)$$

$$\begin{cases} (f_i - f_i')^2 \; if \; features \; are \; numeric \\ 1 \; if \; features \; are \; numeric \; and \; f_i = f_i' \\ 0 \; if \; features \; are \; numeric \; and \; f_i \neq f_i' \end{cases}$$

که در آن w_i وزن ویژگی iام است و مقدار آن بین ۰ و ۱ می‌باشد. همچنین $\delta = 0.0001$ یک عدد ثابت کوچک است که در فرمول قرار داده می‌شود تا مانع از تقسیم بر صفر شود.

فاصله منهتن که به‌عنوان فاصله بلوک شهری[8] نیز شناخته می‌شود یک نوع از فاصله اقلیدسی است که شیوه محاسبه آن در معادله (۳) نشان داده شده است.

$$sim(p.p') = \frac{1}{\left[\sum_{i=1}^{n} w_i Dis(f_i.f_i')+\delta\right]} \quad \delta = 0.0001$$

$$\begin{cases} |f_i - f_i'| \; if \; features \; are \; numeric \\ 1 \; if \; features \; are \; numeric \; and \; f_i = f_i' \\ 0 \; if \; features \; are \; numeric \; and \; f_i \neq f_i' \end{cases}$$

۲-۲ K نزدیک‌ترین همسایه[9] (KNN)

الگوریتم k نزدیک‌ترین همسایه (KNN) نمونه‌ای از یادگیری مبتنی بر نمونه است که در آن دادگان آموزشی ذخیره شده و سپس طبقه‌بندی برای یک رکوردی که هنوز طبقه‌بندی نشده است به سادگی با مقایسه آن با شبیه‌ترین رکوردها در مجموعه داده آموزشی انجام می‌شود. KNN به‌عنوان یک پارامتر حیاتی بسیار موثر بر دقت شناخته می‌شود. در انتخاب مقدار k، اگر k خیلی کوچک باشد تاثیر داده‌های پرت یا نوفه‌های غیرعادی زیاد می‌شود و اگر مقدار k خیلی بزرگ باشد آنگاه رفتار محلی مورد نظر نادیده گرفته می‌شود. اگرچه برخی از مطالعات پیشنهادهایی برای ثابت بودن مقدار k دارند، اما در اکثر مقالات این مقدار متغیر می‌باشد [۱۰].

۲-۳- تابع پاسخ

در این گام مشخص خواهد شد که چگونه می‌توان

4- Historical database
5- Similarity function
6- K-nearest neighbors
7- Solution function
8- City block distance
9- K Nearest Neighbors

از پروژه‌های مشـــابه شناسایی شده در گام قبلی استفاده نمود تا بتوان هزینه پروژه جدید را به شکلی مناسب تخمین زد. روش‌های ارزیابی مشابه‌ترین پروژه[10]، میانگین بدون وزن، میانه و میانگیـــن وزن‌دار فاصله معکوس[11] به‌عنوان پایه‌ای برای تابع پاسخ توسط بسیاری از محققان استفاده شـــده‌اند. میانگیـــن به‌عنوان معیاری کلاســـیک شـــناخته می‌شـــود که گرایش بـــه مرکـــز دارد. همچنیـــن میانگین هزینه‌های نرم‌افزاری را زمانی می‌توان محاســـبه نمود که در آن k>1 باشـــد. میانه، معیار کلاســـیک دیگری است که گرایـــش به مرکز دارد. در این معیار هزینه‌های نرم‌افزاری را زمانی می‌توان محاســـبه کرد که در آن k>2 باشد. میانه نســـبت به میانگین معیار قوی‌تری است زیرا میانه نسبت به داده‌های پرت حســـاس است و داده‌های پرت با افزایش تعـــداد پروژه‌ها افزایش می یابـــد. میانگین وزن‌دار فاصله معکوس نشان‌دهنده این است که پروژه‌های مشابه اهمیت بیشتری نســـبت به پروژه‌های کمتر مشابه دارند. فرمول میانگین وزن‌دار فاصله معکوس در معادله (٤) نشان داده شده است [١١].

$$\widehat{C_p} = \sum_{k=1}^{n} \frac{Sim(P.P_k)}{\sum_{i=1}^{n} Sim(P.P_k)} C_{pk} \qquad (٤)$$

که در آن P نشـــان‌دهنده پروژه‌ای است که باید هزینه آن تخمین‌زده شود. P_k نشـــان‌دهنده kامین پروژه مشابه است. $Sim(P.P_k)$ نشـــان‌دهنده شباهت بین پروژه‌های P و P_kاست و C_{pk} نشـــان‌دهنده هزینه مشابه‌ترین پروژه به P_k است.

۳- کارهای مرتبط

تاکنون روش‌های مختلفی در راستای تخمین مبتنی بر تشابه ارائه شده است که تمرکز همگی آن‌ها بر بهبود دقت تخمین بوده اســـت؛ زیرا مهم‌ترین مســـئله‌ای که در حوزه تخمین هزینه‌های نرم‌افزار با آن مواجه هستیم، عدم وجود تخمین با دقت مناســـب می‌باشـــد. برای حل این مشـــکل، اســـتفاده از انتخاب ویژگی و وزن‌دهی ویژگی‌ها بیشترین روش‌هایی هستند که به آن‌ها پرداخته شده است.

بـــرای افزایش دقت تخمین توســـعه نرم‌افزار در روش مبتنی بر تشـــابه، از روش‌های کلاسیک [١٢] و روش‌های یادگیری ماشـــین [١٣, ١٤] به وفور اســـتفاده شده است. اســـتفاده از الگوریتم‌های بهینه‌سازی، به‌خصوص هوش ازدحامی برای افزایش دقت تخمین توسعه نرم‌افزار توسط بســـیاری از محققین پیشنهاد شده اســـت [١٥, ١٦]. ون و همکارانش[12] در یک بررســـی سیســـتماتیک ادبیات ٨ مدل یادگیری ماشین را برای تخمین تلاش شناسایی کردند که می‌توانند به بهبود دقت تخمین کمک کنند [١٧].

در رابطه با افزایش دقت در روش مبتنی بر تشـــابه تا کنون الگوریتم‌های مختلفی مورد اســـتفاده قرار گرفته‌اند. در روش مبتنی بر تشـــابه معمولی هر کدام از ویژگی‌های پروژه، مستقل هستند و میزان تاثیر مشابهی دارند. آئور و همکارانش[13] [١٩] برای تخمین دقیقتر پیشنهاد دادند که هر کدام از ویژگی‌ها دارای میزان تاثیر متفاوتی باشـــند. برای وزن‌دهی به ویژگی‌ها، برخی از روش‌های کلاســـیک مانند رگرسیون [٢٠] مورد استفاده قرار گرفته‌اند.

اگرچه، بیشـــتر تحقیقـــات در این زمینه با اســـتفاده از محاســـبات نـــرم و الگوریتم‌های فراابتکاری انجام شـــده اســـت [٢١-٢٤]. در این بین، از سیســـتم های فازی [٢٥]، الگوریتم‌های تکاملی و شبکه‌های عصبی مصنوعی [٣, ٢٦] نیز برای تعدیل روش مبتنی بر تشابه استفاده شده است. آزه و همکاران[14] [٢٧] به ارزیابی روش‌های تطبیقی مبتنی بر تشـــابه پرداختند و به این نتیجه رسیدند که روش‌های تنظیم خطی می‌توانند پاسخ بهتری را تولید کنند.

علیرغم کارهای مناســـبی که در این حوزه انجام شده است، محققان نرم‌افزاری باز هم اشتیاق مناسبی در بهبود تخمین نرم‌افزار دارند و هرگونه راهکار جدیدی که احتمال تخمین بهتری حاصل کند را جدی می‌گیرند. به همین دلیل، در این مطالعه، ســـعی شده است، از مدل تکاملی یادگیر در جهت بهبود تخمین اندازه نرم‌افزار بهره‌گیری شود.

10- Closet analogy
11- Inverse distance Weighted Mean
12- Wen et al.
13- Auer et al.
14- Azzeh et al.

۴- مدل تکاملی یادگیر (LEM)

تمام روش‌های رایج محاسبات تکاملی از اصول تئوری داروین الهام گرفته شده‌اند و تحت این قانون می‌باشند: «قانون کلی منجر به پیشرفت همه موجودات عالی شده است، این است که قوی‌ترها زنده بمانند و ضعیف‌ترها بمیرند». محاسبات تکاملی داروین نیمه‌کورکورانه هستند. در آن‌ها برای تولید جمعیت جدید از عملگرهایی مانند جهش (یک تولیدمثل غیرمعمول با تنوع) بازترکیبی (بازسازی و تکثیرجنسی) و انتخاب (بقای مطلوب) استفاده می‌شود. در این نوع تکامل افراد جمعیت جدید با استفاده از افراد آموزش دیده جمعیت‌های قبلی هدایت نمی‌شود بلکه یک فرآیند آزمون و خطا است که به‌صورت موازی انجام می‌شود. ایده اصلی مدل تکاملی یادگیر از ترکیب روش‌های جستجوی تکاملی و یادگیری ماشین الهام گرفته شده است [۲۸]. این راهکار، یک مدل یادگیری ماشین را اجرا می‌کند تا بتواند افرادی از جمعیت که در انجام وظایف از بقیه بهتر عمل می‌کنند را تشخیص دهد. این دلایل به‌عنوان فرضیه‌های استنتاجی بیان می‌شوند و سپس برای تولید جمعیت جدید این افراد مورد استفاده قرار می‌گیرند. در واقع، این مدل جمعیت جدید را با استفاده از فرضیه‌های مربوط به افراد با تناسب بالا در جمعیت گذشته، تولید می‌کند.

مدل تکاملی یادگیرممکن است بین حالت یادگیری ماشین و حالت تکامل داروینی جابه‌جا شود و یا به‌طور کامل به یادگیری ماشین تکیه کند. این الگوریتم اگر تنها حالت یادگیری ماشین را اجرا کند آنقدر آن را به‌صورت تکراری اعمال می‌کنند تا یادگیری ماشین برای جمعیت جدید به تکامل برسد. هنگام کار با هر دو حالت، این الگوریتم زمانی که توانست شرط خروج را برآورده سازد از یک حالت به حالت دیگر انتقال می یابد. فرایند تکامل تا زمانی ادامه پیدا می‌کند که جواب راه‌حل برای این الگوریتم رضایت‌بخش باشد و یا منابع اختصاص یافته خسته شوند.

روند تکامل درالگوریتم تکاملی یادگیر با تعدادی از اعضا از یک جمعیت اولیه آغاز می‌شود. در پردازش‌های تکاملی اعضا ممکن است راه‌حل‌های مسائل و یا دستورالعمل‌هایی تولید راه‌حل‌ها باشند در اینجا یک فرم کلی از فرایند الگوریتم تکاملی یادگیر بیان می‌شود:

۱. ایجاد جمعیت: جمعیت اولیه به‌صورت تصادفی یا براساس یک روش خاص ساخته می‌شود.

۲. اجرای حالت یادگیری ماشین:

الف) استخراج اکسترما: از اعضای جمعیت حاضر دو گروه انتخاب می‌کنیم یک گروه با کارایی بالا که به‌طور خلاصه به آن H-group می‌گوییم در گروه دیگر با کارایی پایین را L-group می‌نامیم. مقدار این گروه‌ها با استفاده از مقدار تابع تناسب به‌دست می‌آید.

ب. ایجاد فرضیه‌ها: یک روش یادگیری ماشین برای توصیف H-group و L-group اعمال می‌کنیم که بتواند این دو گروه را از یکدیگر متمایز کند. هنگام یادگیری توصیف H-group (یا L-group) باید به تاریخچه تکامل یعنی جمعیت گذشته و یا توصیف جمعیت‌های گذشته نیز توجه کرد.

پ. ایجاد یک جمعیت جدید: برای تولید یک جمعیت جدید نمونه‌های یاد گرفته شده در H-group با جمعیت جدید ترکیب می‌شود. توصیف نمونه‌ها یا به‌صورت تصادفی یا به‌صورت قانون‌هایی از نمونه‌های توصیف شده انجام می‌شود.

ت. برو به مرحله الف و حالت یادگیری ماشین را تکرار کن تا زمانی که به شرط خاتمه برسد. هنگامی که شرط خاتمه در حالت یادگیری ماشین برآورده شد می‌توان یکی از اقدامات زیر را انجام داد:

۱. فرایند تکامل را خاتمه داد.

۲. تکرار فرایند از مرحله ۱ که به این عملیات شروع مجدد از اول گفته می‌شود.

۳. اجرای حالت تکامل داروینی: در این قسمت یکی از روش‌های تکامل داروینی را اجرا می‌کنیم. به این معنا که یک جهش بازترکیبی و انتخاب را برای تولید جمعیت جدید اعمال می‌کنیم. این عملیات را تا زمانی انجام می‌دهیم که حالت تکامل داروین به شرط خاتمه برسد.

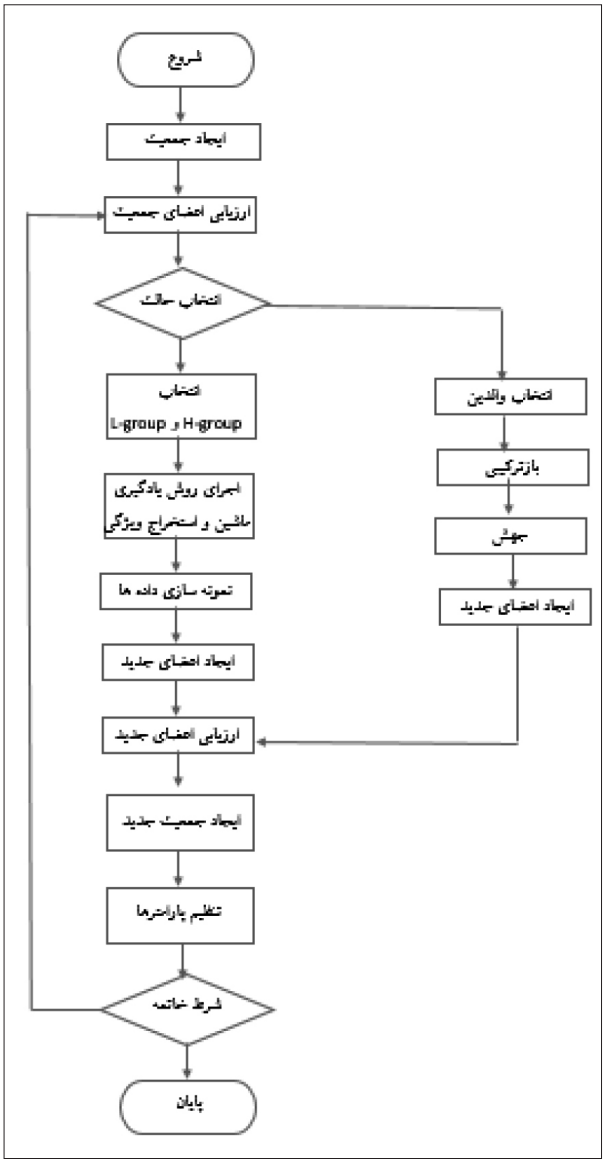
٤. تبادل کردن حالت‌هـــا: در این حالت به مرحله ۲ می رویم سپس انتقال بین مراحل ۲ و ۳ را آنقدر انجام می‌دهیم که شرط خاتمه آن برآورده شود.

شـــرط خاتمه در حالت یادگیری ماشـــین در صورت رســـیدن به یک سطح از کارایی به‌دست می‌آید. اگر در این مرحله شرط خاتمه الگوریتم تکاملی یادگیر هنوز برآورده نشده باشد، عملیات شروع مجدد را اجرا می‌کند یا به شرط خاتمه حالت تکامـــل داروینی می‌رود. اگر الگوریتم تکاملی یادگیر همیشه عملیات شـــروع مجدد را انتخاب کند، روند تکامل تنها براســـاس یک برنامه تکراری از حالت یادگیری ماشین می‌شود. چنین نسخه‌ای از عملیات الگوریتم تکاملی یادگیـــر را uniLEM می‌نامیم زیرا حالت تکامل داروینی را شامل نمی‌شود. در صورتی که الگوریتم تکاملی یادگیر از هر دو حالت استفاده نمایند، به آن duoLEM گفته می‌شود. در شکل ۱ روندنمای الگوریتم الگوریتم تکاملی یادگیر نشان داده شده است.

وقتی الگوریتم تکاملی یادگیر عملیات شـــروع مجدد را انتخـــاب می‌کند، باید به مرحله ۱ برود. یک راه‌حل ســـاده برای اجرای مرحله ۱ این است که مانند دیگر الگوریتم‌های تکاملی به‌صورت تصادفی یک جمعیت جدید را تولید کنیم. اما در این الگوریتم این اتفاق معمولاً در هنگام اجرای مرحله ۱ برای اولین بار اتفاق می‌افتد و برای اجرای عملیات شروع مجدد روش‌های دیگری از جمله انتخاب نخبگان، اجتناب از شکست‌های گذشته، استفاده از توصیه‌ها و ایجاد یک نوع دیگر وجود دارد.

۵- مدل پیشنهادی

در این تحقیق ســـعی شـــده اســـت که مدلی مبتنی بر الگوریتـــم تکاملی یادگیر در جهت ارائه تخمین مناســـب‌تر در پروژه‌هـــای نرم‌افزاری ارائه گردد. مدل پیشـــنهادی از دو مرحله آموزش و آزمون تشـــکیل شده است. جزئیات معمـــاری ایـــن مراحـــل در بخش‌هـــای بعدی ارائه شـــده است.



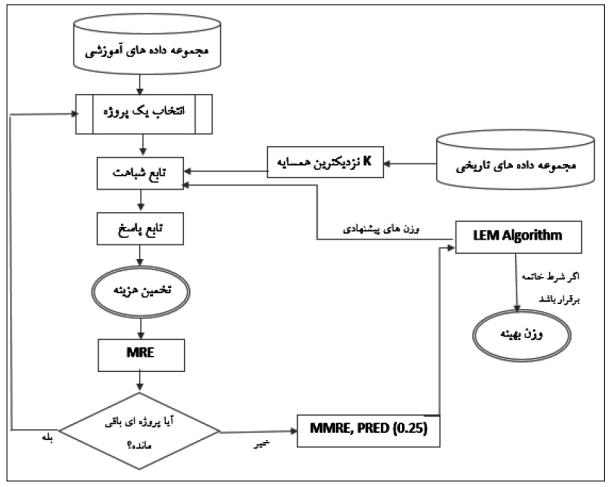
شکل ۱: روندنمای الگوریتم تکاملی یادگیر

۵-۱ مرحله آموزش

در ایـــن مرحلـــه مجموعه‌ای از داده‌های آموزشـــی به مدل ارائه می‌شود و سیستم تخمین هزینه مبتنی بر تشابه توسط وزن ویژگی‌ها تنظیم می‌شـــود و الگوریتم تکاملی یادگیر برای به حداقل رســـاندن خطاها به بررســـی بردار وزن در نمونه‌های آموزشی می‌پردازد. در شکل ۲ معماری این مرحله نشان داده شده است.

۵-۲ مرحله آزمون

در این مرحله چارچوب آموزش دیده شده با داده‌های آزمون برای پیش‌بینی هزینه ارائه می‌شـــود. در شـــکل ۳



شکل۲ : معماری سیستم داده‌های آموزشی

معماری این مرحله نشان داده شده است.

۶ - راه‌اندازی آزمایش

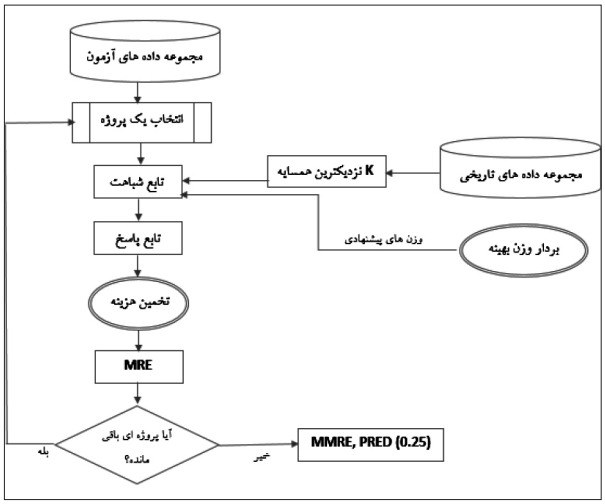
۶-۱- معیار های ارزیابی

از آنجا که پژوهش‌های مختلف، توابع متفاوتی را برای آزمایش روش خود مورد استفاده قرار می‌دهند، انتخاب معیارهای ارزیابی مناسب جهت مقایسه با روش‌های دیگر کاری مشکل است. زیرا در مقالات متفاوت معیارهای ارزیابی به کار گرفته شده، متفاوت بوده و در اکثریت موارد کد منبع برنامه در دسترس نیست. به همین دلیل در این تحقیق سعی شده است تا از عمومی‌ترین معیارهای ارزیابی که در اکثریت قریب به اتفاق مقالات مورد استفاده قرار گرفته شده است، استفاده شود.

شاخص عملکردی که معمولاً برای اندازه‌گیری کارایی مدل‌های پیش‌بینی نرم‌افزار استفاده می‌شود اندازه خطای نسبی[۱۵] (MRE) است که درصد خطای مطلق را نسبت به تلاش واقعی محاسبه می‌کند و در معادله ٥ نشان داده شده است. میانگین MRE ها را MMRE می‌نامیم که در معادله ٦ نشان داده شده است. همچنین میانه MREها، MdMRE نامیده می‌شود.

$$MRE_i = \frac{|Estimated\ value - actual\ value|}{actual\ value} \quad (٥)$$

15- Magnitude of relative error



شکل ۳: معماری سیستم داده‌های آزمون

$$MMRE = \frac{\sum_{i=1}^{N} MRE_i}{N} \quad (٦)$$

شاخص دیگری که وجود دارد شاخص عملکرد PRED(X) است که درصد پیش‌بینی‌هایی که مقدار x را درست تشخیص می‌دهند، را نشان می‌دهد در معادله (٧) تعریف می‌شود.

$$PRED(x) = \sum_{i=1}^{N} D_i * \frac{100}{N} \quad (٧)$$

$$when\ x = 25\ the\ PRED\ metric\ is\ defind$$

$$as\ PRED(0.25)\ D_i = \begin{cases} 1\ if\ MMRE < \frac{x}{100} \\ 0\ otherwise \end{cases}$$

به دلیل این‌که اکثریت مقالات برای ارزیابی روش‌های پیشنهادی خود از این سه معیار استفاده نموده‌اند، در این تحقیق نیز از این معیارها استفاده شده است.

۶-۲ مجموعه داده‌ها

در این تحقیق برای ارزیابی روش پیشنهادی از ۲ مجموعه داده تخمین نرم‌افزار استفاده شده است که عبارتند از:

- Desharnais: این مجموعه داده شامل داده‌های پروژه‌های نرم‌افزاری کانادایی است.
- Maxwell: این مجموعه داده شامل داده‌های پروژه‌های نرم‌افزاری فنلاند است.

در ادامه در جدول شماره ۱ لیستی از آمار توصیفی مجموعه داده‌ها فهرست شده است.

جدول ۱: آمار توصیفی مجموعه داده‌ها

نام مجموعه داده	تعداد ویژگی‌ها	تعداد پروژه‌ها	مشخصات مجموعه داده				
			کمترین مقدار	بیشترین مقدار	میانگین	میانه	چولگی
Desharnais	۱۲	۸۱	۵۴۶	۲۳۹۴۰	۵۰۴۶	۳۶۴۷	۲
Maxwell	۲۷	۶۲	۵۸۳	۶۳۶۹۴	۸۲۲۳,۲	۵۱۸۹,۵	۳,۲۶

۶-۳ راه اندازی آزمایش

برای طراحی آزمایش و ارزیابی روش پیشنهادی با استفاده از آزمایش‌های مختلف ابتدا باید عملیات پیش‌پردازش بر روی داده‌ها اجرا شود که برای این کار با استفاده از معادله min-max داده‌ها در بازه صفر تا یک نرمال خواهند شد. عملیات نرمال‌سازی به جهت حذف تاثیرات متفاوت ویژگی‌ها انجام می‌شود. پس از آن برای آموزش مدل پیشنهادی باید مجموعه داده‌های آموزشی را به دو مجموعه داده‌های آموزشی و داده‌های آزمون تقسیم‌بندی نمود که در این تحقیق، این کار را با استفاده از روش 10-fold cross validation انجام شده است. در این تقسیم‌بندی مجموعه داده‌ها به‌صورت تصادفی به ۱۰ مجموعه تقریباً برابر تقسیم می‌شود. از این قسمت‌ها، یک قسمت به‌عنوان مجموعه داده آزمون و ۹ قسمت دیگر به‌عنوان مجموعه داده‌های آموزشی در نظر گرفته می‌شود.

همانطور که قبلا بیان شد روش تخمین مبتنی بر تشابه دارای سه پارامتر کنترلی شامل تابع شباهت، k نزدیک‌ترین همسایه و تابع پاسخ می‌باشد. در طراحی این آزمایش، جهت تابع شباهت از فواصل اقلیدوسی و منهتن و برای به‌دست آوردن مقدار تخمین پیشنهادی از رایج‌ترین توابع پاسخ که میانه و میانگین می‌باشند، استفاده شده است. همچنین تعداد نزدیک‌ترین پروژه به یکدیگر را به‌صورت متغیر و در بازه ۱ تا ۵ فرض شده‌اند. پس از مشخص نمودن مقادیر پارامترها، ابتدا تأثیر ترکیب‌های مختلف این پارامترها را در فضای جستجوی هر پارامتر تعریف می‌کنیم و در انتها مناسب‌ترین مدل تولید شده را جهت آزمایش با دیگر روش‌ها انتخاب می نماییم.

۷- نتایج علمی

۷-۱ آزمایش روش پیشنهادی

برای آزمایش روش پیشنهادی به بررسی امکان به‌دست آوردن مطلوب‌ترین پیکربندی روش تخمین مبتنی بر تشابه پرداخته می‌شود. برای انجام این کار از سه معیار ارزیابی PRED، MMRE(۰,۲۵) و MdMRE بر روی مجموعه داده‌های Desharnais و Maxwell استفاده شده‌اند. نتایج مربوطه به ترتیب در جداول ۲و ۳ نشان داده شده است و مقادیر قابل توجه پررنگ شده‌اند.

براساس مدل پیشنهادی، ارزیابی بهترین انتخاب در معیارهای مختلف متفاوت است. اما مناسب‌ترین مدلی که در هر دو مجموعه داده مناسب باشد و بتواند به خوبی پاسخگو باشد مربوط به فاصله اقلیدوسی و سه نزدیک‌ترین همسایه و همچنین تابع پاسخ میانه می‌باشد. همچنین کمترین مقدار MMRE و MdMRE مربوط به فاصله اقلیدسی با ۵ نزدیک‌ترین همسایه و نیز تابع پاسخ میانه است ولی مقدار PRED آن‌ها با این که مقادیر نسبتا مناسبی دارد اما بیشترین مقدار در میان مدل‌ها را شامل نمی‌شود. بر طبق نتایج به‌دست آمده بیشترین مقدار PRED مربوط به فاصله منهتن و سه نزدیک‌ترین همسایه و تابع پاسخ میانه می‌باشد.

۷-۲ مقایسه کارایی روش پیشنهادی با سایر روش‌ها

برای مقایسه چند روش یکی از مواردی که باید رعایت شود این است که معیار کیفیت توابع ارزیابی و بازه‌های انتخاب شده مجموعه داده‌ها یکسان باشد تا بتوان به درستی مقایسه را انجام داد. در این تحقیق میزان کارآیی پیشنهادی (New Model) با دیگر الگوریتم‌های تکاملی از جمله ژنتیک (GA)، تکامل تفاضلی (DE) و الگوریتم ازدحام

جدول ۲: نتایج حاصله بر روی مجموعه داده Desharnais

Similarity	k	Solution	Training Set			Testing Set		
			MMRE	PRED	MdMRE	MMRE	PRED	MdMRE
	K=1	CA	0.25	0.56	0.16	0.34	0.50	0.26
Euclidean	K=2	mean	0.41	0.47	0.40	0.40	0.48	0.39
	K=3	mean	0.39	0.43	0.38	0.37	0.46	0.38
		median	0.24	0.55	0.17	0.31	0.50	0.25
	K=4	mean	0.24	0.55	0.17	0.21	0.60	0.15
		median	0.28	0.50	0.21	0.27	0.51	0.20
	K=5	mean	0.24	0.55	0.17	0.26	0.50	0.20
		median	0.42	0.21	0.50	0.46	0.21	0.54
			0.23	0.57	0.12	0.21	0.61	0.12
Manhattan	K=1	median						
	K=2	mean	0.36	0.44	0.36	0.35	0.47	0.36
	K=3	mean	0.38	0.36	0.40	0.37	0.32	0.35
		median	0.48	0.40	0.65	0.46	0.43	0.65
	K=4	mean	0.42	0.25	0.48	0.53	0.20	0.62
		median	0.25	0.54	0.17	0.24	0.55	0.17
	K=5	mean	0.36	0.44	0.30	0.34	0.46	0.28
		median	0.35	0.44	0.30	0.37	0.46	0.32
			0.27	0.55	0.22	0.28	0.50	0.23

جدول۳: نتایج حاصله بر روی مجموعه داده Maxwell

similarity	k	solution	Training Set			Testing Set		
			MMRE	PRED	MdMRE	MMRE	PRED	MdMRE
	K=1	CA	0.47	0.46	0.35	0.48	0.46	0.36
Euclidean	K=2	mean	0.66	0.54	0.48	0.56	0.62	0.45
	K=3	mean	0.64	0.46	0.34	0.90	0.38	0.60
		median	0.77	0.58	0.35	0.11	0.61	0.16
	K=4	mean	0.50	0.50	0.30	0.49	0.50	0.30
		median	0.51	0.48	0.29	0.51	0.52	0.29
	K=5	mean	0.66	0.6	0.45	0.56	0.68	0.42
		median	0.67	0.50	0.30	0.62	0.54	0.29
			0.50	0.50	0.30	0.10	0.51	0.12
Manhattan	K=1	median						
	K=2	mean	0.7	0.4	0.47	0.71	0.36	0.49
	K=3	mean	0.62	0.52	0.47	0.53	0.62	0.42
		median	0.64	0.46	0.34	0.90	0.38	0.60
	K=4	mean	0.65	0.62	0.45	0.55	0.63	0.43
		median	0.59	0.52	0.36	0.61	0.44	0.37
	K=5	mean	0.54	0.58	0.33	0.55	0.58	0.33
		median	0.66	0.38	0.34	0.62	0.46	0.31
			0.54	0.6	0.30	0.55	0.68	0.31

ذرات (PSO) مقایسه شده و تا امکان مقایسه عملکرد آن‌ها فراهم شود. زیرا این الگوریتم‌ها ساختاری مشابه با الگوریتم پیشنهادی دارند. نتایج این مقایسات که بر روی مجموعه داده‌های مختلف پیاده‌سازی شده، در جداول ۵ و ٦ نشان داده شده است.

همان‌طور که در جداول فوق نمایش داده شده است،

جدول ۴: بهترین نتایج ارزیابی مدل براساس دو مجموعه داده

تابع شباهت	K	تابع پاسخ	معیار
اقلیدسی	۳	میانه	بهترین ارزیابی (در نظر گرفتن هر سه معیار)
اقلیدسی	۵	میانه	کمترین مقدار MMRE و MdMRE
منهتن	۳	میانه	بیشترین مقدار PRED

جدول ۵: مقایسه مدل پیشنهادی با الگوریتم‌های DE و PSO در مجموعه داده Desharnais

Model	Training Set			Testing Set		
	MMRE	PRED	MdMRE	MMRE	PRED	MdMRE
New Model	۰,۷۷	۰,۵۸	۰,۳۵	۰,۱۱	۰,۶۱	۰,۱۶
PSO-ABE	۰,۳۳	۰,۳۰	۰,۲۶	۰,۳۰	۰,۶۱	۰,۲۶
DABE	۰,۶۳	۰,۵۸	۰,۲۳	۰,۳۲	۰,۵۲	۰,۳۵
GA-ABE	۰,۶۰	۰,۵۸	۰,۳۰	۰,۲۸	۰,۵۹	۰,۱۷

مقدار معیار های MMRE و MdMRE کمتر از مقادیر مشابه آن‌ها نسبت به دیگر الگوریتم‌ها می‌باشد. این مسئله در هر دو مجموعه داده صدق می‌کند. این مسئله به‌طور واضح‌تر در شکل‌های ٤ و ٥ نشان داده شده است.

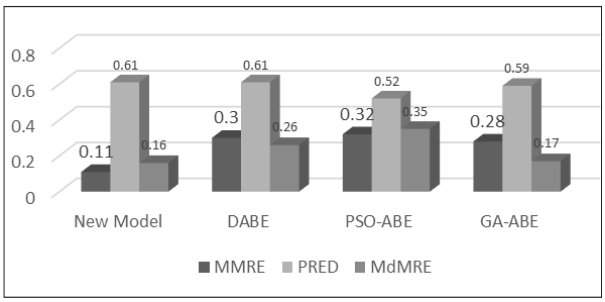
چنانچه در شکل‌های ٤ و ٥ نشان داده شده است، روش پیشنهادی توانسته است نتایج بهتری را تولید کند و می‌تواند به‌عنوان راهکاری برای تخمین بهتر اندازه نرم‌افزار مورد استفاده قرار گیرد.

۸- نتیجه گیری

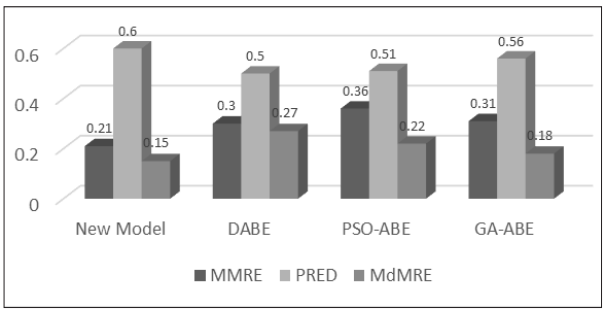
بهبود تخمین اندازه نرم‌افزار همواره به‌عنوان یکی از حوزه‌های جذاب در مهندسی نرم‌افزار مورد توجه محققان بوده است. در این مطالعه به جهت افزایش دقت تخمین هزینه در نرم‌افزار، روش بهینه‌سازی وزن براساس مدل تکاملی یادگیر در روش مبتنی بر تشابه پیشنهاد شده است. به دلیل این‌که مدل تکاملی یادگیر اعضای جمعیت جدید را توسط یک سری از فرآیندها تولید نموده و فرضیه‌ها را ایجاد می‌نماید، می‌تواند مشکل تصادفی بودن تولید جمعیت در الگوریتم‌های مبتنی بر مدل داروین را تا حدودی مرتفع

جدول۶ : مقایسه مدل پیشنهادی با الگوریتم‌های DE و PSO در مجموعه داده Maxwell

Model	Training Set			Testing Set		
	MMRE	PRED	MdMRE	MMRE	PRED	MdMRE
New Model	0.24	0.55	0.17	0.21	0.60	0.15
PSO-ABE	0.31	0.5	0.27	0.30	0.50	0.27
DABE	0.33	0.52	0.22	0.36	0.51	0.22
GA-ABE	0.33	0.55	0.24	0.31	0.56	0.18



شکل ۴: نمودار بررسی مقدار معیارهای MMRE و MdMRE و PRED در مجموعه داده Maxwell



شکل ۵: نمودار بررسی مقدار معیارهای MMRE و MdMRE و PRED در مجموعه داده Desharnais

سازد و همچنین سرعت همگرایی را بالا ببرد. به همین دلیل، در این مطالعه، این الگوریتم به‌عنوان پایه‌ای برای راهکار جدید در تخمین اندازه نرم‌افزار مورد استفاده قرار گرفت. راهکار پیشنهادی بر روی مجموعه داده‌های مختلف و در حالت های متفاوت مورد آزمایش قرار گرفت. مطالعات تجربی در این مطالعه نشان می‌دهد که در بیشتر موارد، نتایج حاصل از راهکار پیشنهادی در معیارهای ارزیابی متفاوت و در مقایسه با دیگر الگوریتم‌های تکاملی از جمله ژنتیک، تکامل تفاضلی و الگوریتم ازدحام ذرات رضایت بخش بوده است.

مراجع

[1]S. Keaveney and K. Conboy, "Cost estimation in agile development projects," in ECIS, 2006, vol. 169, pp. 183-197.

[2]M. Jorgensen and M. Shepperd, "A systematic review of software development cost estimation studies," IEEE Transactions on software engineering, vol. 33, no. 1, 2007.

[3]Y.-F. Li, M. Xie, and T. Goh, "A study of the non-linear adjustment for analogy based software cost estimation," Empirical Software Engineering, vol. 14, no. 6, pp. 603-643, 2009.

[4]V. K. Bardsiri, D. N. A. Jawawi, S. Z. M. Hashim, and E. Khatibi, "A PSO-based model to increase the accuracy of software development effort estimation," Software Quality Journal, vol. 21, no. 3, pp. 501-526, 2013.

[5]A. Tosun, B. Turhan, and A. B. Bener, "Feature weighting heuristics for analogy-based effort estimation models," Expert Systems with Applications, vol. 36, no. 7, pp. 10325-10333, 2009.

[6]S. H. S. Moosavi and V. K. Bardsiri, "Satin bowerbird optimizer: A new optimization algorithm to optimize ANFIS for software development effort estimation," Engineering Applications of Artificial Intelligence, vol. 60, pp. 1-15, 2017.

[7]J. Wojtusiak and R. S. Michalski, "The LEM3 implementation of learnable evolution model and its testing on complex function optimization problems," in Proceedings of the 8th annual conference on Genetic and evolutionary computation, 2006, pp. 1281-1288.

[8]C. Cobos, D. Estupiñán, and J. Pérez, "GHS+ LEM: Global-best Harmony Search using learnable evolution models," Applied Mathematics Computation, vol. 218, no. 6, pp. 2558-2578, 2011.

[9]D. T. Larose and C. D. Larose, Discovering knowledge in data: an introduction to data mining. John Wiley & Sons, 2014.

[10]N.-H. Chiu and S.-J. Huang, "The adjusted analogy-based software effort estimation based on similarity distances," Journal of Systems and Software, vol. 80, no. 4, pp. 628-640, 2007.

[11]G. Kadoda, M. Cartwright, L. Chen, and M. Shepperd, "Experiences using case-based reasoning to predict software project effort," in Proceedings of the EASE 2000 Conference, Keele, UK, 2000: Citeseer.

[12]E. Mendes, N. Mosley, and S. Counsell, "A replicated assessment of the use of adaptation rules to improve Web cost estimation," in 2003 International Symposium on Empirical Software Engineering, 2003. ISESE 2003. Proceedings., 2003, pp. 100-109: IEEE.

[13]T. R. Benala and R. Bandarupalli, "Least square support vector machine in analogy-based software development effort estimation," in 2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE), 2016, pp. 1-6: IEEE.

[14]T. R. Benala and R. Mall, "DABE: Differential evolution in analogy-based software development effort estimation," Swarm Evolutionary Computation, vol. 38, pp. 158-172, 2018.

[15]Q. Liu, X. Chu, J. Xiao, and H. Zhu, "Optimizing non-orthogonal space distance using pso in software cost estimation," in 2014 IEEE 38th Annual Computer Software and Applications Conference, 2014, pp. 21-26: IEEE.

[16]S. M. Satapathy, B. P. Acharya, and S. K. Rath, "Early stage software effort estimation using random forest technique based on use case points," IET Software, vol. 10, no. 1, pp. 10-17, 2016.

[17]J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," Information Software Technology, vol. 54, no. 1, pp. 41-59, 2012.

[18]M. Azzeh, D. Neagu, and P. I. Cowling, "Fuzzy grey relational analysis for software effort estimation," Empirical Software Engineering, vol. 15, no. 1, pp. 60-90, 2010.

[19]M. Auer, A. Trendowicz, B. Graser, E. Haunschmid, and S. Biffl, "Optimal project feature weights in analogy-based cost estimation: Improvement and limitations," IEEE Transactions on Software Engineering, vol. 32, no. 2, pp. 83-92, 2006.

[20]A. B. Nassif, M. Azzeh, A. Idri, and A. Abran, "Software development effort estimation using regression fuzzy models," Computational intelligence and neuroscience, vol. 2019, 2019.

[21]S.-J. Huang and N.-H. Chiu, "Optimization of analogy weights by genetic algorithm for software effort estimation," Information software technology, vol. 48, no. 11, pp. 1034-1045, 2006.

[22]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.

[23]P. S. Kumar and H. Behera, "Role of soft computing techniques in software effort estimation: an analytical study," in Computational Intelligence in Pattern Recognition: Springer, 2020, pp. 807-831.

[24]P. S. Kumar, H. S. Behera, A. Kumari, J. Nayak, and B. Naik, "Advancement from neural networks to deep learning in software effort estimation: Perspective of two decades," Computer Science Review, vol. 38, p. 100288, 2020.

[25]M. Azzeh, D. Neagu, and P. Cowling, "Improving analogy software effort estimation using fuzzy feature subset selection algorithm," in Proceedings of the 4th international workshop on Predictor models in software engineering, 2008, pp. 71-78.

[26]M. Azzeh, "A replicated assessment and comparison of adaptation techniques for analogy-based effort estimation," Empirical Software Engineering, vol. 17, no. 1-2, pp. 90-127, 2012.

[27]M. Azzeh, A. B. Nassif, and L. L. Minku, "An empirical evaluation of ensemble adjustment methods for analogy-based effort estimation," Journal of Systems and Software, vol. 103, pp. 36-52, 2015.

[28]R. S. Michalski, "Learnable evolution model: Evolutionary processes guided by machine learning," Machine learning, vol. 38, no. 1-2, pp. 9-40, 2000.