ارائه یک مدل جدید جهت تخمین تلاش لازم برای توسعه سرویس‌های نرم افزاری

عمید خطیبی‌بردسیری[۱،*]، سید محسن هاشمی[۲]، محمدرضا رزازی[۳]

اطلاعات مقاله

دریافت مقاله: ۱۳۹۳/۰۵/۲۳

پذیرش مقاله: ۱۳۹۴/۰۹/۰۷

واژگان کلیدی:

تخمین تلاش،

سرویس نرم‌افزاری،

روش قیاس،

الگوریتم تکامل تفاضلی،

مدل وزن‌دهی.

چکیده

تخمین دقیق تلاش لازم برای توسعه سرویس‌های نرم افزاری یک چالش بزرگ هم در صنعت و هم برای محققین است. مفهوم تلاش یک پارامتر مهم و تأثیرگذار در فرآیند توسعه و مدیریت سرویس‌های نرم افزاری است. تخمین دقیق تلاش به مدیران پروژه کمک می‌کند تا منابع را بهتر تخصیص دهند و هزینه و زمان را طوری مدیریت کنند که پروژه در وقت و بودجه تعیین شده به اتمام برسد. یکی از مشهورترین روش‌های تخمین تلاش، استفاده از قیاس و مقایسه یک سرویس با موارد مشابه قبلی است. متأسفانه روش قیاس بدون استفاده از وزن‌های مناسب و ارزش‌دهی به ویژگی‌های یک سرویس، نتایج خوبی نخواهد داشت. بنابراین در این مقاله سعی شده تا با ترکیب روش قیاس و الگوریتم تکامل تفاضلی یک مدل کارا و قابل اطمینان برای برآورد تلاش لازم جهت توسعه سرویس‌های نرم افزاری ایجاد شود. مدل پیشنهادی بر روی داده‌های واقعی مستخرج از پایگاه داده ISBSG و دو پایگاه داده مصنوعی مورد ارزیابی قرار گرفت و نتایج با روش‌های مشهور تخمین تلاش مقایسه گردید؛ مقادیر به دست آمده برای مخازن داده‌ای ISBSG، همگن و ناهمگن به ترتیب و به طور میانگین بهبود ۲۸٪، ۳۴٪ و ۱۹٪ را نشان می‌داد.

۱- مقدمه

تحویل به موقع و در بودجه تعیین شده یک سرویس، یکی از نگرانی‌های اصلی اکثر شرکت‌های نرم‌افزاری است. تلاش لازم برای توسعه یک سرویس نرم افزاری[۲] جزء مهم‌ترین و تأثیرگذارترین پارامترهای یک پروژه است و به سه دلیل اصلی تغییرات زیاد دنیای سخت افزار، تغییرات زیاد درخواست‌های مشتریان و تنوع زیاد سرویس‌های نرم‌افزاری تخمین این پارامتر مشکل می‌باشد [۱]. واضح است که

*. پست الکترونیک نویسنده مسئول: a.khatibi@srbiau.ac.ir

۱. دانشجو دکتری نرم‌افزار، گروه مهندسی کامپیوتر، دانشکده فنی و مهندسی، دانشگاه آزاد اسلامی، واحد علوم و تحقیقات تهران

۲. استادیار گروه مهندسی کامپیوتر، دانشکده فنی و مهندسی، دانشگاه آزاد اسلامی، واحد علوم و تحقیقات تهران

۳. دانشیار گروه کامپیوتر و فناوری اطلاعات، دانشکده مهندسی، دانشگاه صنعتی امیرکبیر، پلی‌تکنیک تهران

[2] Software service development effort

تخمین کمتر یا بیش از حد، هر دو موجب اتلاف منابع و به خطر افتادن موقعیت کمپانی می‌شوند. با توجه به اینکه فرآیند تخمین باید در فازهای ابتدایی پروژه انجام شود نیاز به یک روش مطمئن است که با اطلاعات کم و ناقص اولیه بتواند کار کند. برای یک سرویس نرم‌افزاری ویژگی‌های مختلفی وجود دارد که مدل‌های تخمین تأثیر جنبه‌های گوناگون مثل تجربه تیم توسعه، اندازه سرویس، سطح دانه-بندی[1] و پیچیدگی سرویس را بر میزان تلاش توسعه نشان می‌دهند [۲, ۳]. تاکنون روش‌های مختلفی برای تخمین تلاش پیشنهاد شده است که می‌توان همه آنها را در پنج گروه اصلی جای داد:

آ- روش‌های پارامتری همانند COCOMO[2]، SLIM[3] و SERR-SEM[4] [۴, ۵]

ب- قضاوت کارشناسانه[5] مثل روش‌های Delphi و WBS[6] [۶]

ج- مدل‌های مبتنی بر یادگیری مثل شبکه‌های عصبی، درخت تصمیم و روش قیاس [۷-۹]

د- متدهای رگرسیون همانند MLR[7] و SWR[8] [۱۰]

ه- مدل‌های ترکیبی مانند LMES[9] [۲].

روش‌های الگوریتمی برای تخمین، از مفاهیم آماری و معادلات ریاضی استفاده می‌کنند؛ در مقابل روش‌های غیرالگوریتمی بیشتر مبتنی بر تجزیه و تحلیل داده‌های تاریخی و قبلی هستند [۱۱]. ایدهٔ اولیه روش‌های الگوریتمی با قانون استفاده از تعداد خطوط کد، در سال ۱۹۵۰ آغاز شد [۱۲]. در سال ۱۹۸۱، بَری بوهم یک مدل جدید به نام COCOMO را برای تخمین تلاش توسعه نرم‌افزارها ارائه داد که از معادلات تجربی برای تخمین استفاده می‌کرد [۱۳]. مدل‌های دیگر مثل SLIM و SEER-SEM هم اصول روش COCOMO را ادامه دادند. معرفی نقطه کارکرد[10] توسط آلبِرشت در سال ۱۹۸۳ یکی از اتفاقات مهم در حوزل اندازه‌گیری نرم‌افزار بود که امکان اندازه‌گیری را در مراحل اولیهٔ پروژه به مدیران می‌داد و از تأثیرات منفی روش قبلی یعنی خطوط کد تا حدی زیادی جلوگیری می‌کرد [۱۴]. تغییر زیاد در متدلوژی‌های توسعه نرم‌افزار و پیشرفت در روش‌های تخمین، منجر به توسعه نسخهٔ جدیدی از مدل COCOMO یعنی COCOMO II در سال ۲۰۰۰ توسط بوهم شد [۴]. از سوی دیگر به دلیل عدم توانایی روش‌های الگوریتمی در مهار کردن رفتار پویای پروژه‌های نرم‌افزاری و نبود اطلاعات کامل از یک پروژه در مراحل اولیه، متدهای غیر الگوریتمی ارائه شدند. روش قضاوت کارشناسانه که در سال ۱۹۶۳ ارائه شد نمونه‌ای از این نوع روش‌ها بود [۶]. در این رویکرد افراد خبره و متخصص تخمین خود را دربارهٔ مقدار تلاش تا رسیدن به یک توافق نهایی به اشتراک می‌گذارند. درخت تصمیم[11] روش دیگری از گروه متدهای غیر الگوریتمی است که با ساخت یک درخت با استفاده از پروژه‌های کامل شده قبلی و ویژگی‌های آنها، میزان تلاش را در برگ‌های درخت به دست می‌آورد [۱۵]. در این میان مشهورترین روش تخمین غیر الگوریتمی، روش قیاس[12] می‌باشد که در سال ۱۹۹۷ توسط شِپرد و اِسکوفیلد مطرح شد [۱۶]. این متد برای تخمین تلاش از مقایسه یک پروژه با موارد مشابه قبلی استفاده می‌کند و عمل مقایسه براساس ویژگی‌های مستقل دو پروژه انجام می‌شود. در این مقاله با تمرکز بر روش قیاس به عنوان متد پایه و الگوریتم تکامل تفاضلی[13] به عنوان ابزار کمکی، یک مدل وزن‌دهی پویا برای تخمین دقیق‌تر مقدار تلاش ارائه خواهیم کرد. ادامهٔ این مقاله در هشت قسمت سازمان‌دهی شده است: بخش دوم به مرور کارهای مرتبط می‌پردازد. بخش سه و چهار به ترتیب الگوریتم تکامل تفاضلی و روش قیاس را توضیح می‌دهند. مدل پیشنهادی در بخش پنجم آورده شده و بخش شش

[1] Granularity
[2] COnstructive COst Model (COCOMO)
[3] Software LIfecycle Management (SLIM)
[4] Software Evaluation and Estimation of Resources-Software Estimating Model (SEER-SEM)
[5] Expert judgment
[6] Work Breakdown Structure (WBS)
[7] Multiple Linear Regression (MLR)
[8] Step-Wise Regression (SWR)
[9] Localized Multi-EStimator (LMES)
[10] Function Point (FP)
[11] Decision tree
[12] Analogy Base Estimation (ABE)
[13] Differential Evolution (DE)

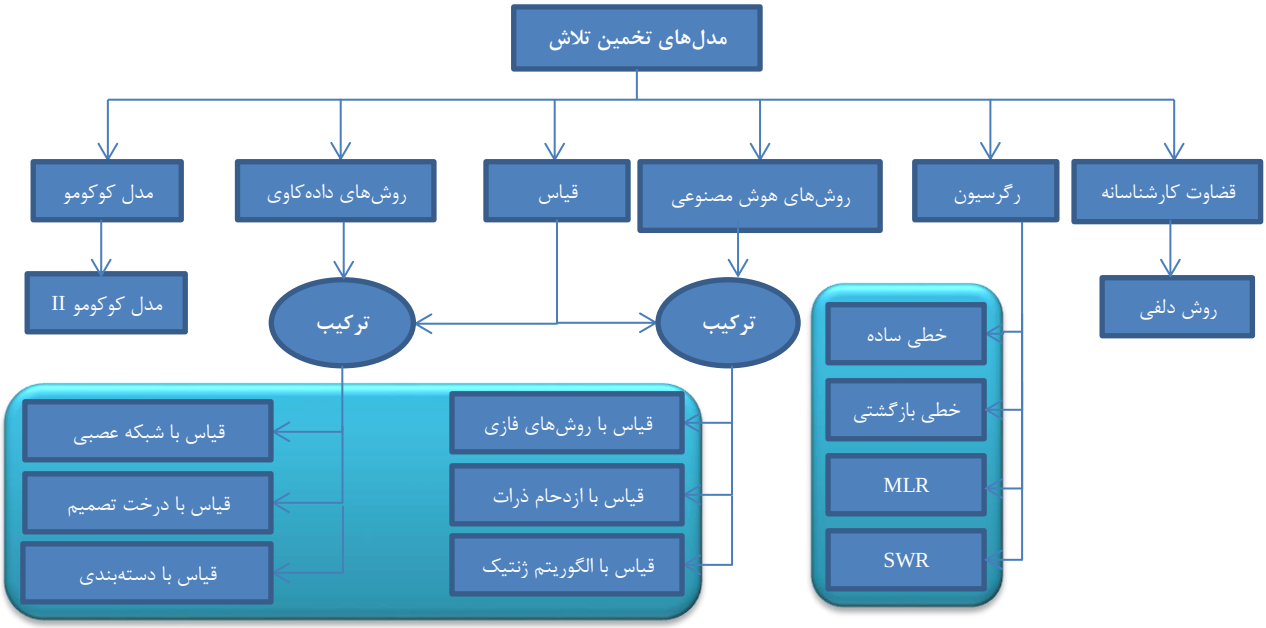
و هفت نیز به ترتیب نحوهٔ ارزیابی و نتایج به دست آمده را نشان می‌دهند. در نهایت بخش هشتم به نتیجه‌گیری و پیشنهاد کارهای آینده اختصاص دارد.

۲- کارهای مرتبط

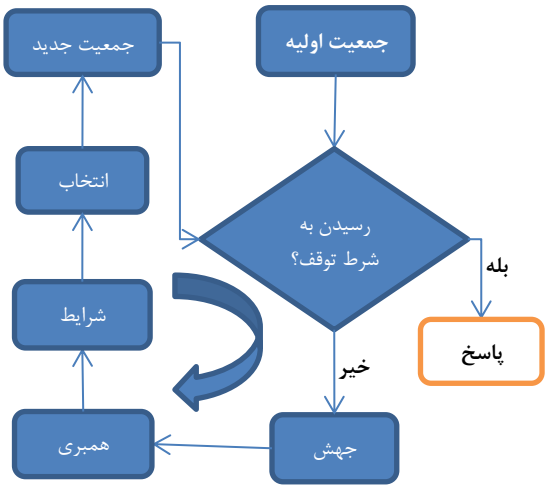
انواع مختلف روش‌های تخمین تلاش، سال‌ها مورد تحقیق و ارزیابی قرار گرفتند. این روش‌ها از فرضیات و معادلات ساده شروع شده و هم اکنون به تکنیک‌های پیچیده‌ای رسیده‌اند. به طور کلی روش‌های غیر الگوریتمی و مبتنی بر تحلیل پروژه‌های تمام شده قبلی، نسبت به روش‌های الگوریتمی کارایی بهتری دارند [۱۱, ۱۷]. انواع مختلف رگرسیون، مدل‌های COCOMO، SLIM و COCOMO II از مشهورترین مدل‌های الگوریتمی هستند و در مقابل روش‌های قیاس، درخت تصمیم [۱۸]، قضاوت کارشناسانه [۶]، شبکه‌های عصبی [۱۹]، قوانین فازی [۲۰] و الگوریتم‌های بهینه‌سازی [۱۷] از موارد معروف روش‌های غیر الگوریتمی می‌باشند. مشهورترین روش غیر الگوریتمی یعنی متد قیاس به طور گسترده‌ای در مطالعات مختلف مورد استفاده قرار گرفته است [۷, ۸, ۲۳-۲۱]. روش قیاس به تنهایی دقت و کارایی بالایی را نسبت به سایر متدهای تخمین از خود نشان نمی‌دهد و بنابراین به صورت کلی محققین علاقه‌مند به ترکیب کردن این روش با سایر تکنیک‌ها هستند [۱۶, ۲۷-۲۴]. شکل ۱ انواع ترکیبات استفاده شده کنونی که از متد قیاس به عنوان روش پایه بهره برده‌اند را نشان می‌دهد. یکی از مهم‌ترین بخش‌ها در روش قیاس تابع شباهت می‌باشد که با استفاده از ویژگی‌های دو پروژه، میزان تشابه آنها را محاسبه می‌کند [۱۶]. بازیابی پروژه‌های مناسب به اندازه‌گیری دقیق و درست میزان تشابه بستگی دارد در غیر این صورت مقایسهٔ نادرستی صورت خواهد گرفت. واضح است که ویژگی‌های مختلف یک سرویس نرم‌افزاری تأثیر یکسانی بر روی میزان تلاش لازم جهت توسعه آن سرویس ندارند. در واقع وزن-دهی ویژگی‌ها و تعیین میزان همبستگی آنها با تلاش توسعه یک مسئله بسیار مهم است. ویژگی‌های مهم‌تر باید وزن بیشتر و ویژگی‌های کم تأثیر بر تلاش، باید وزن کمتری بگیرند و اگر هیچ همبستگی دیده نمی‌شود به طور کلی حذف شوند. تا کنون مطالعات بسیاری بهبود کارایی روش قیاس را به کمک تکنیک‌های وزن‌دهی نشان داده‌اند [۲, ۸, ۱۷, ۲۴-۲۲, ۳۰-۲۷]. الگوریتم ژنتیک یکی از پرکاربردترین روش‌ها در بهبود کارایی روش قیاس است. لی و همکاران با فرمول‌بندی مسئله تخمین تلاش، الگوریتم ژنتیک را برای جستجوی پارامترهای مناسب وزن به کار بردند. نتایج آنها بهبود محسوس دقت روش قیاس را نشان می‌داد [۲۴]. به‌طور مشابه الگوریتم ازدحام ذرات به وسیله خطیبی و همکاران بر روی متد قیاس به کار رفته است. در این تحقیق هم نتایج به دست آمده، تأثیر مثبت جستجو بر کارایی مدل تخمین را تأیید می‌کرد. خطیبی و همکاران با تکیه بر سادگی و سرعت الگوریتم ازدحام ذرات نشان دادند کارایی این روش از الگوریتم ژنتیک بیشتر است [۱۷]. در مطالعات پیشین هیچ کاربردی از الگوریتم تکامل تفاضلی دیده نمی‌شود امّا ویژگی‌های خاص این الگوریتم و سازگاری آن با خواص متد قیاس، احتمال تأثیرگذاری بالای آن را نشان می‌دهد. سرعت همگرایی، قابل تنظیم بودن، محاسبات آسان و وجود پیکربندی‌های مختلف از نقاط قوت الگوریتم تکامل تفاضلی است که آن را برای ترکیب با متد قیاس مناسب می‌سازد.

۳- الگوریتم تکامل تفاضلی

این الگوریتم یکی از جدیدترین روش‌های جستجو است که فرضیهٔ آن برخواسته از تلاش‌های پرایس در زمینه حل مسئله تطبیق چندجمله‌ای‌ها بود. این مسئله توسط استورن مطرح شد و در نتیجه تلاش برای حل آن، الگوریتم تکامل تفاضلی در سال ۱۹۹۶ ارائه گردید [۳۱]. الگوریتم تکامل تفاضلی یک روش مبتنی بر جمعیت است و همانند الگوریتم ژنتیک با تکامل جمعیت اولیه به سمت هدف پیش می‌رود. خلاصه‌ای از مراحل این الگوریتم به صورت فلوچارت در شکل ۲ آورده شده است.



شکل ۱- انواع مختلف مدل‌های مستقل و ترکیبی تخمین تلاش



شکل۲- مراحل مختلف الگوریتم تکامل تفاضلی

سه پارامتر کنترلی در این الگوریتم وجود دارد: آ. ثابت تفاضلی[1] (نرخ جهش[2]) F ب. ثابت همبری[3] C_r ج. اندازه جمعیت[4] NP. تفاوت این روش با الگوریتم ژنتیک در نحوه همگرایی و پیشروی به سمت هدف است؛ در اینجا در هر تکرار برای هر فرد از جمعیت، پنج مرحله بعدی در نظر گرفته می‌شود [۳۲]:

جهش: در این مرحله انتخاب تصادفی سه اندیس (r_1، r_2 و r_3) از جمعیت صورت می‌گیرد؛ به طوری که دوبه‌دو متفاوت[5] با یکدیگر و همچنین متفاوت با اندیس جاری (i) باشند. سپس بردار جهش طبق معادله (۱) (بسته به شِمای انتخابی برای الگوریتم) ایجاد می‌شود [۳۰].

$$v_{i,G+1} = x_{r_1,G} + F.\left(x_{r_2,G} - x_{r_3,G}\right) \qquad (۱)$$

همبری: پس از کامل شدن فرآیند جهش، در این مرحله فرد مورد امتحان[6] ایجاد می‌شود. هر بُعد از فرد مورد امتحان مطابق معادلهٔ (۲) با احتمال همبری C_r، ژن جهش یافته را به ارث خواهد برد. در این معادله $U_{i,G+1}$ فرد مورد امتحان (کاندید جایگزینی در نسل بعد) را نشان می‌دهد و بازهٔ تغییر j بین ۱ تا D است که در آن متغیر D تعداد ابعاد مسئله می‌باشد. پارامتر k نیز به طور تصادفی در همین محدوده انتخاب می‌شود و این گزینش تضمین می‌کند که حداقل یکی از متغیرهای مسئله تغییر خواهد کرد [۳۰].

$$U_{j,i.G+1} = \begin{cases} V_{j,i.G+1} & \text{اگر } rand_j \le C_r \text{ یا } j = k \\ X_{j,i.G} & \text{در غیر این صورت} \end{cases} \qquad (۲)$$

[1] Differential constant
[2] Mutation rate
[3] Cross over
[4] Population size
[5] Mutual exclusion
[6] Trial individual

بازبینی محدودیت‌های مرزی: اگر پس از تغییر مقادیر، برخی پارامترهای فرد مورد امتحان محدودیت‌های مسئله را نقض کنند، در این مرحله مطابق فرمول (۳) برگشتن آنها به محدودهٔ مجاز انجام خواهد شد. دو متغیر $X_{j,min}$ و $X_{j,max}$ به ترتیب میزان مجاز برای حد پایین و بالای بعد jام را نشان می‌دهند [۳۱].

$$U_{j,i} = X_{j,min} + (X_{j,max} - X_{j,min}).rand_{ji}[0,1) \qquad (۳)$$

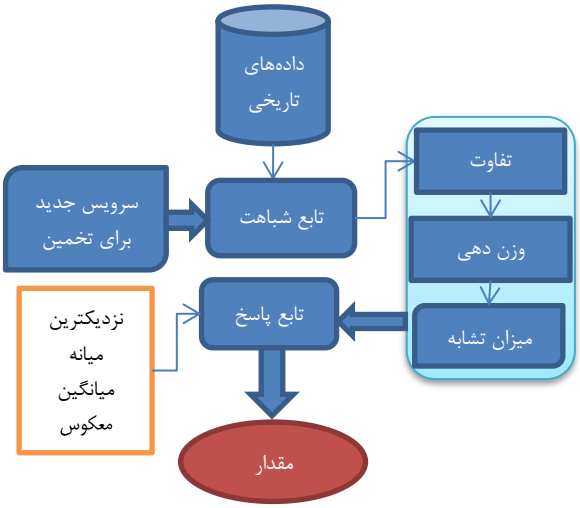
انتخاب: ابتدا مقادیر دو تابع برازش[1] مربوط به فرد جاری و فرد مورد امتحان، به دست آمده و با هم مقایسه می‌شوند. اگر مقدار شایستگی فرد مورد امتحان کمتر یا برابر با فرد جاری باشد (برای کمینه کردن تابع برازش)، فرد امتحانی جایگزین فرد جاری در جمعیت جدید می‌گردد در غیر این صورت هیچ تغییری در نسل قبلی ایجاد نمی‌شود. معادلهٔ (۴) نحوهٔ انجام فرآیند انتخاب را نشان می‌دهد [۳۰].

$$X_{i.G+1} = \begin{cases} U_{i.G+1} & \text{آنگاه } U \text{ مقدار برازش بهتر} \\ X_{i.G} & \text{آنگاه } X \text{ مقدار برازش بهتر} \end{cases} \qquad (۴)$$

آزمون توقف: سرانجام زمانی که الگوریتم کامل شد (بسته به شرط توقف الگوریتم که می‌تواند تعداد نسل‌ها و یا میزان اختلاف ارزش دو نسل باشد)، بهترین فرد[2] جمعیت به عنوان پاسخ الگوریتم انتخاب می‌شود. اساس کار این الگوریتم بر مبنای عملیات ساده ریاضی (جمع، تفریق، ضرب و...) بدون استفاده از مشتق بر روی بردارهاست؛ به همین دلیل این روش نسبت به متدهای مشابه، از سادگی و سرعت همگرایی[3] بالایی برخوردار است [۳۳-۳۱].

۴- تخمین مبتنی بر قیاس

ساختار کلی و مراحل روش تخمین مبتنی بر قیاس در شکل ۳ مشاهده می‌شود. در واقع در این روش تک‌تک ویژگی‌های دو سرویس با هم مقایسه شده و فاصله‌ای به عنوان تفاوت دو سرویس تعریف می‌شود؛ این فاصله از طریق معادلهٔ (۵) به دست می‌آید (در حالت فاصلهٔ اقلیدسی[4]) و به آن تابع شباهت[5] می‌گویند.



شکل ۳- مراحل مختلف روش تخمین مبتنی بر قیاس

در این معادله s و s' دو سرویس نرم‌افزاری مورد مقایسه و w_i وزن ویژگی iام است. وزن هر ویژگی سرویس بسته به اهمیت آن، می‌تواند بین ۰ و ۱ متغیر باشد. همچنین f_i و f'_i، iامین ویژگی هر سرویس را نشان می‌دهند و مقدار n تعداد کل ویژگی‌ها است. به جهت اینکه مخرج کسر هیچ‌گاه صفر نشود از ثابت δ با مقدار ۰,۰۰۰۱ استفاده می‌شود [۱۶].

$$Sim(s,s') = \frac{1}{\left[\sqrt{\sum_{i=1}^{n} w_i Dis(f_i,f_i')} + \delta\right]} \qquad (۵)$$

$$Dis(f_i,f_i') = \begin{cases} (f_i - f_i')^2 & \text{ویژگی‌های عددی} \\ 0 & f_i = f_i' \text{ و غیر عددی} \\ 1 & f_i \neq f_i' \text{ و غیر عددی} \end{cases}$$

$Dis(f_i , f'_i)$ نشان‌دهنده فاصله و اختلاف ویژگی‌های دو سرویس است؛ هرچه فاصله بیشتر شود شباهت دو سرویس به هم کمتر خواهد شد. نکته مهم آن است که قبل از مقایسه دو سرویس باید همه ویژگی‌ها نرمال سازی شوند تا تغییرات یکنواخت در میزان تلاش مورد مقایسه به دست آید. برای اندازه‌گیری میزان شباهت فرمول‌های مختلفی از جمله فاصله اقلیدسی (فرمول شماره ۳)، فاصله منهتن[6]

[1] Fitness function
[2] Best individual
[3] Convergence speed

[4] Euclidean distance
[5] Similarity function
[6] Manhatan distance

[۱۶] و مقدار گری[1] [۲۰] پیشنهاد شده است. تابع دیگر روش قیاس، تابع پاسخ[2] است که برای تخمین تلاش لازم جهت توسعه سرویس از میزان تلاش سرویس‌های مشابه (پیشنهادهای تابع شباهت در مرحله قبل) استفاده می‌کند. مشهورترین توابع پاسخ عبارت‌اند از نزدیک‌ترین، میانه، میانگین و معکوس. منظور از نزدیک‌ترین، شبیه‌ترین سرویس به مورد مقایسه است. تابع میانگین، میانگین مقادیر تلاش‌های به دست آمده از K سرویس مشابه را در نظر می‌گیرد وقتی که $K>1$ است. میانه، مقدار میانه تلاش سرویس‌ها را با فرض $K>2$ به دست می‌آورد و در نهایت تابع معکوس میزان تلاش را بر اساس معادلهٔ (۶) محاسبه می‌کند [۲, ۱۷].

$$E_p = \sum_{k=1}^{K} \frac{Sim(s, s_k)}{\sum_{i=1}^{K} Sim(s, s_i)} E_{s_k} \qquad (۶)$$

در این فرمول، Es_k تلاش توسعه سرویس S_k، K تعداد سرویس‌های مشابه و E_p تلاش نهایی سرویس نرم‌افزاری مورد تخمین است. در واقع استفاده از مقادیر و توابع مختلف در روش قیاس، پیکربندی‌های مختلفی را به وجود می‌آورد که بالطبع کارایی‌های مختلفی خواهند داشت. در قسمت نتایج تجربی انواع این مقادیر مورد بررسی قرار خواهند گرفت.

۵- مدل پیشنهادی

مدل پیشنهادی در این مقاله با استفاده از ترکیب دو روش قیاس و تکامل تفاضلی سعی در تخمین تلاش لازم برای توسعه یک سرویس نرم‌افزاری دارد. روش کار به این شکل است که الگوریتم تکامل تفاضلی کمک می‌کند تا دقت انتخاب و تعیین میزان تشابه در تابع شباهت بالاتر رود. در واقع الگوریتم این کار را با وزن‌دهی فرمول (۵) عنوان شده در بخش چهارم انجام می‌دهد. در مدل پیشنهادی دو مرحله اصلی وجود دارد:

آ. مرحلهٔ آموزش[3]: در این مرحله به ساخت مدل و وزن-دهی ویژگی‌ها پرداخته خواهد شد.

ب. مرحلهٔ آزمون[4]: این مرحله به ارزیابی و آزمودن مدل طراحی شده اختصاص دارد.

۵-۱- معیارهای ارزیابی

یک هدف اصلی و در واقع مهم‌ترین پارامتر برای مقایسه مدل‌های مختلف تخمین تلاش، میزان دقت یک مدل است یعنی مقدار تخمین زده شده[5] تا چه حد از مقدار واقعی[6] فاصله دارد. در این مقاله کارایی مدل‌ها، با استفاده از مشهورترین و پرکاربردترین متریک‌ها، شامل MRE[7]، MMRE[8] و PRED[9] اندازه‌گیری خواهد شد. موارد ذکر شده به شکل زیر محاسبه می‌شوند [۱۵]:

$$MRE = \frac{|E' - E|}{E} \qquad (۷)$$

$$MMRE = \frac{\sum_{i=1}^{N} MRE}{N} \qquad (۸)$$

$$PRED(X) = \frac{A}{N} \qquad (۹)$$

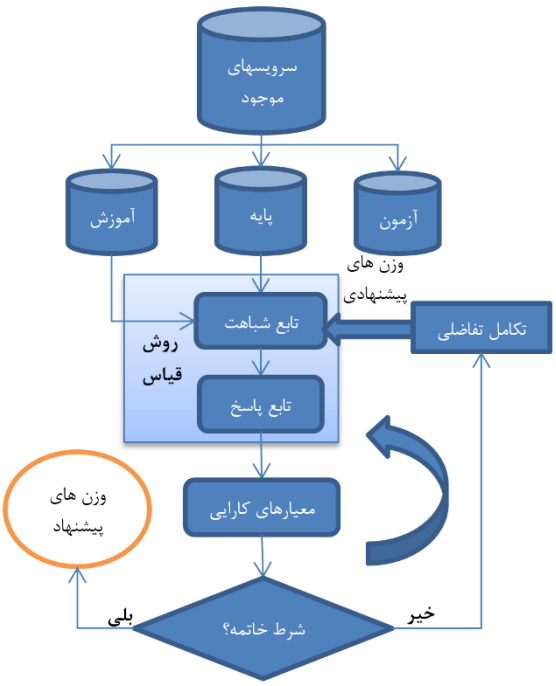
که در آنها E میزان تلاش واقعی، E' تلاش تخمین زده شده، A تعداد سرویس‌ها با MRE کمتر مساوی X و N تعداد کل سرویس‌هاست. مقدار پذیرفته شدهٔ X در تمامی تحقیقات، برابر ۰/۲۵است. همان‌گونه که از تعاریف مشخص است برای همهٔ مدل‌ها، مقدار MRE و MMRE باید کمینه و PRED باید بیشینه باشد. دلیل انتخاب این متریک‌ها، تواتر استفاده در کارهای پیشین و امکان مقایسه صحیح بین مدل‌ها بوده است [۲, ۷, ۸, ۱۷, ۲۷, ۳۴].

۵-۲- مرحله آموزش

شکل ۴ ساختاری مصور از مرحلهٔ آموزش مدل پیشنهادی را نشان می‌دهد. در ابتدا کل سرویس‌های نرم‌افزاری موجود (پایگاه دادهٔ تاریخی) به سه مجموعهٔ مجزای سرویس‌های

[1] Grey value
[2] Solution function
[3] Train stage
[4] Test stage
[5] Estimated values
[6] Real value
[7] Magnitude of Relative Error
[8] Mean Magnitude of Relative Error
[9] Percentage of predictions

پایه[1]، آموزش[2] و آزمون[3] با تعداد مساوی یا نزدیک به مساوی (بسته به اندازهٔ پایگاه داده) تقسیم می‌شوند.



شکل ۴ - مرحله آموزش مدل پیشنهادی

از دو مجموعهٔ پایه و آموزش در مرحله آموزش مدل و از مجموعه آزمون و پایه به صورت مشترک در مرحلهٔ آزمون استفاده خواهد شد. در مرحلهٔ آموزش تک‌تک سرویس‌های موجود در مجموعهٔ آموزش برداشته شده و با وزنی که الگوریتم تکامل تفاضلی به تابع شباهت می‌دهد کل فرآیند قیاس را طی می‌کنند. بعد از اتمام این فرآیند، یک مقدار MMRE و PRED(0.25) کلی برای مجموعهٔ آموزش به دست می‌آید که الگوریتم از آن، برای تنظیم وزن‌های خود استفاده می‌کند. در واقع تابع برازش الگوریتم تکامل تفاضلی مقدار MMRE-PRED(0.25) خواهد بود و همان‌گونه که از تعاریف مشخص است باید در حداقل مقدار ممکن خود باشد. در تکرار بعد همین مقادیر این بار با وزن‌های پیشنهادی جدید به دست خواهند آمد و این روند به تعداد مشخص انجام خواهد شد تا مدل بر پایه روش قیاس و وزن‌های الگوریتم تکامل تفاضلی ساخته شود. همان‌گونه که در شکل نیز مشخص است در مرحلهٔ آموزش دو چرخه وجود دارد که اولی مربوط به محاسبهٔ MMRE برای سرویس‌های آموزش است و دومی برای تنظیم وزن‌ها به‌وسیلهٔ الگوریتم تکامل تفاضلی (این چرخهٔ داخلی با جزییات کامل در شکل (۲) آورده شده است). ورودی این مرحله سرویس‌های آموزش و پایه و خروجی آن وزن‌های مناسب خواهد بود که خود به عنوان ورودی مرحلهٔ بعد (آزمون) به کار می‌رود.

۵-۳- مرحله آزمون

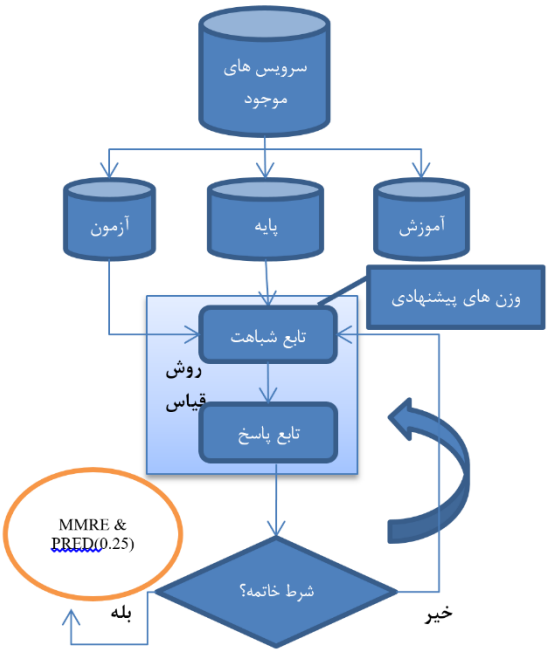
هدف اصلی مرحله آزمون، ارزیابی دقت مدل ساخته شده مرحله قبل، به‌وسیلهٔ سرویس‌های جدید است. در این مرحله تک‌تک سرویس‌های آزمون انتخاب شده و پس از طی مراحل قیاس و به کمک وزن‌های پیشنهادی مرحلهٔ قبل، مقدار MRE خود را پیدا می‌کنند. در نهایت پس از آزمودن تمام سرویس‌ها مقدار کلی پارامترهای کارایی به دست می‌آید. ساختاری مصور از این مرحله در شکل ۵ نمایش داده شده است. از آنجا که سه مجموعه آزمون، آموزش و پایه کاملاً مجزا و مساوی انتخاب شدند و از مجموعهٔ آزمون در مرحله قبل هیچ استفاده‌ای نشده، تمامی سرویس‌های مورد آزمون نامرئی و دیده نشده‌اند؛ از این رو می‌توان به صحت و درستی نتایج ارزیابی اطمینان کامل داشت. از مرحلهٔ قبل برداری از وزن‌ها در محدودهٔ صفر تا یک و به تعداد ویژگی‌های سرویس داریم. این بردار که خروجی الگوریتم تکامل تفاضلی و براساس تابع برازش تعریف شده است در این مرحله بر روی داده‌های آزمون اعمال می‌شود. انتظار این است که به کمک این وزن‌ها، روش قیاس دقت و کارایی بیشتری از خود نشان دهد و تخمین تلاش توسعه سرویس دقیق‌تر شود.

۶- فرآیند ارزیابی

استفاده از سرویس‌هایی که در مرحلهٔ ساخت یک مدل تخمین استفاده شده‌اند در فرآیند ارزیابی آن، منجر به نتایج مصنوعی و خوش‌بینانه‌ای خواهد شد. در این حالت خطای

[1] Base services
[2] Train services
[3] Test services

تخمین به صورت غیر واقع‌بینانه کاهش یافته و آزمایش، عملکرد واقعی مدل را نشان نمی‌دهد. بنابراین یک مدل تخمین حتماً باید با سرویس‌های جدید و دیده نشده ارزیابی گردد. در همین زمینه استفاده از تکنیک اعتبارسنجی متقاطع[1] یک ارزیابی دقیق و واقعی‌تر را در پی خواهد داشت [۳۵]. در این تکنیک تمامی داده‌ها به چند دسته مجزای آموزش و آزمون تقسیم می‌شوند که سرویس‌های آزمون به عنوان سرویس‌های دیده نشده عمل می‌کنند و این فرآیند تا آزمون همه سرویس‌ها تکرار می‌شود. ادامه این بخش به توصیف داده‌های مورد استفاده و تنظیمات اولیه روش‌های تخمین تلاش مختلف می‌پردازد.



شکل ۵- مرحله آزمون مدل پیشنهادی

۶-۱- توصیف مخزن داده‌ها[2]

برای ارزیابی مدل‌ها و متدهای تخمین نیاز به داده‌های آماری داریم.. در کارهای گذشته مروری بر روی این منابع داده‌ای وجود دارد که ویژگی‌ها و صحت هر یک را مورد بررسی قرار داده است [۳۶]. در این مقاله نیز از یک پایگاه دادهٔ واقعی و دو پایگاه دادهٔ مصنوعی[3] برای ارزیابی مدل‌های تخمین تلاش استفاده خواهیم کرد.

۶-۱-۱- پایگاه داده ISBSG

ISBSG[4] یک کمپانی بزرگ جمع‌آوری داده‌های حوزه فناوری اطلاعات است که در استرالیا قرار دارد [۳۷]. در این مقاله از داده‌های موجود در این منبع استفاده شده که حاوی ۵۰۵۲ پروژهٔ نرم‌افزاری کامل شده در گذشته است. ISBSG از ۱۰۹ ویژگی مستقل برای توصیف هر پروژه استفاده می‌کند و اطلاعات خود را از ۲۴ کشور مختلف دنیا جمع‌آوری کرده است. داده‌های داخل این منبع طیف مختلفی از برنامه‌ها، معماری‌ها، پلتفرم‌ها، زبان‌های برنامه‌سازی، ابزارها و متدهای توسعه را در بر می‌گیرد. در این مقاله زیرمجموعهٔ مناسبی از این پایگاه با رعایت فیلترهای زیر انتخاب شده است:

- از بین داده‌های مختلف، فقط از نرخ کیفی[5] a و b استفاده شده که به گزارش ISBSG در صحت آنها هیچ شکی نیست.
- از میزان تلاش نرمال شده[6] برای مقایسه و بررسی پروژه‌ها استفاده شده است (برای پروژه‌های نرم‌افزاری ناقص و ناتمام).
- به منظور کامل بودن اطلاعات، از پروژه‌هایی که اطلاعات فیلدهای انتخابی مقاله را نداشتند، صرف‌نظر کرده‌ایم.
- از نرخ نرمال‌سازی[7] بزرگ‌تر از ۱،۲ استفاده نشده، که پیشنهاد خود کمپانی ISBSG برای دقت بیشتر است.
- محیط توسعه وب در نظر گرفته شده تا مفهوم سرویس نرم‌افزاری و ارائه خدمت مبتنی بر وب در آن جای گیرد.
- روش اندازه‌گیری همه سرویس‌ها یکسان و IFPUG[8] انتخاب شده است (روش شمارش سرویس).
- فقط تلاش توسعهٔ یک سرویس در نظر گرفته شده است نه مواردی چون ساخت پلتفرم، آموزش کاربر و غیره.

[1] Cross validation technique
[2] Dataset description
[3] Artificial dataset
[4] International Software Benchmarking Standards Group (ISBSG)
[5] Quality rate
[6] Normalized effort
[7] Normalization rate
[8] International Function Point Users' Group (IFPUG)

- از بین همه ویژگی‌های یک سرویس، تنها شش ویژگی مهم و تأثیرگذار بر مقدار تلاش توسعه انتخاب شده‌اند. در نهایت با اعمال فیلترهای بالا جمعاً ۶۶ سرویس نرم‌افزاری به دست آمده است و بر روی این مجموعه کار دنبال خواهد شد. جدول ۱ اطلاعات آماری مربوط به سرویس‌های این پایگاه داده را نشان می‌دهد.

جدول ۱- اطلاعات آماری پایگاه داده ISBSG

متغیر	میانگین	میانه	بیشینه	کمینه	انحراف
تعداد ورودی	۱۶۹	۹۵	۱۱۸۵	۳	۱۹۹
تعداد خروجی	۱۴۳	۶۷	۶۹۸	۱۰	۱۶۵
تعداد پرس‌وجو	۱۵۰	۱۱۶	۶۵۳	۳	۱۳۷
تعداد فایل	۱۲۹	۱۰۸	۳۸۴	۷	۹۷
تعداد واسط	۷۶	۴۳	۴۹۷	۵	۹۵
تعداد FP	۶۷۲	۵۰۷	۲۲۴۵	۱۰۷	۵۳۴
میزان تلاش	۶۸۶۰	۴۸۹ ۹	۶۰۸۲۶	۵۶۲	۸۴۰۶

۶-۱-۲- پایگاه داده‌های مصنوعی

متأسفانه بیشتر پایگاه داده‌های واقعی، قدیمی و کوچک هستند و اکثراً دارای مقادیر خارج از محدودهٔ[1] (نقاط پرت) بسیار می‌باشند. مزیت استفاده از پایگاه داده‌های مصنوعی این است که اولاً اندازهٔ آنها را می‌توان تعیین و بزرگ کرد ثانیاً ویژگی‌ها و خواص آنها توسط کاربر قابل تنظیم است. بدون از دست دادن کلیت موضوع، در این مقاله از روش پیکارد برای مدل‌سازی پایگاه داده مصنوعی استفاده خواهد شد [۳۸]. برای دانستن جزییات بیشتر از این روش می‌توان به منابع [۳۹] و [۴۰] مراجعه کرد. برای توصیف خواص پایگاه مصنوعی از سه ویژگی مستقل واریانس، انحراف[2] و نقاط پرت استفاده می‌شود. معادله (۱۰) نحوه مدل‌سازی این نوع پایگاه را نشان می‌دهد.

$$y = 1000 + 6x_1sk + 3x_2sk + 2x_3sk + e_{het} \quad (۱۰)$$

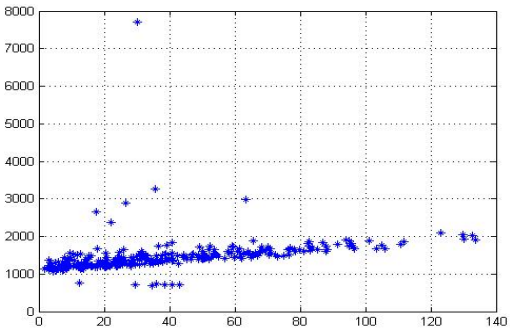
x_1sk ، x_2sk و x_3sk متغیرهای مستقلی هستند که با توزیع تصادفی گامای متغیرهای x_1، x_2 و x_3 با میانگین ۴ و واریانس ۸ ایجاد شده‌اند. توزیع گاما مستقیماً انحراف را می‌سازد و برای مقیاس گذاری متغیرها، x_1 در ۱۰، x_2 در ۳ و x_3 در عدد ۲۰ ضرب می‌شود. عبارت آخر یعنی e_{het} نشان دهنده میزان خطا است و از طریق معادله زیر به دست می‌آید:

$$e_{het} = c \times e \times x_1sk \quad (۱۱)$$

در این عبارت c ضریب ثابت خطا و e متغیر تصادفی با توزیع نرمال میانگین ۰ و واریانس ۱ است. همچنین مقادیر خارج از محدوده از ضرب و تقسیم متغیر وابستهٔ y بر یک ضریب ثابت به دست می‌آیند. در این مقاله یک درصد از کل داده‌ها به عنوان داده‌های پرت در نظر گرفته شده‌اند که نیمی از آنها ضرب و نیم دیگر تقسیم می‌شوند (هر گروه ۰٫۵٪ کل داده‌ها). همچنین با قرار دادن دو مجموعه متفاوت از مقادیر اولیه، به دو پایگاه دادهٔ مصنوعی همگن و ناهمگن رسیده‌ایم که جدول ۲ مجموعهٔ تنظیمات هر یک از آنها را نشان می‌دهد. در ادامه شکل‌های ۶ و ۷ به ترتیب برای ۳۰۰ سرویس نرم‌افزاری مصنوعی بر اساس متغیر x_1 در دو حالت همگن و ناهمگن رسم شده‌اند.

جدول ۲- توصیف پارامترهای پایگاه داده‌های مصنوعی

حالت پایگاه	ضریب نقاط پرت	ضریب میزان خطا
همگن	۲	۰٫۱
ناهمگن	۶	۳



شکل ۶- مقادیر پایگاه داده مصنوعی همگن

۶-۲- تنظیمات اصلی

در این بخش تنظیمات اصلی مربوط به هریک از متدهای تخمین توضیح داده خواهد شد. آماده سازی داده‌ها برای

[1] Outlier

[2] Skewness

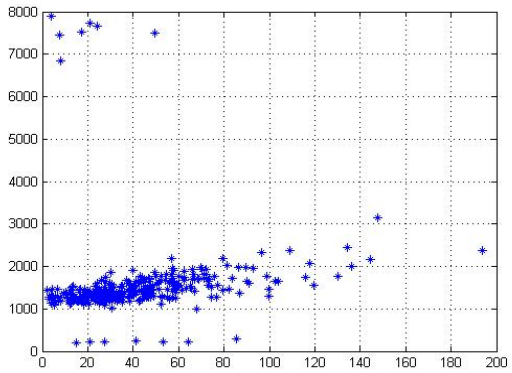
تمام متدهای یکسان است و ویژگی‌های مستقل یک سرویس در بازهٔ صفر تا یک نرمال شده است تا تأثیر یکنواختی بر میزان تلاش توسعه بگذارد. تمامی الگوریتم‌ها ۳۰ بار اجرا و از نتایج به دست آمده میانگین گرفته شده است. اکثر تنظیمات با استفاده از روش آزمون و خطا و در بهترین حالت پاسخ الگوریتم، به دست آمده‌اند. مقادیر مختلف روش قیاس (تابع شباهت، تابع پاسخ و تعداد سرویس مشابه[1]) به کار گرفته شده و مجموعاً ۲۴ ساختار مختلف این متد ارزیابی شده است.

روش قیاس: در این آزمایش‌ها برای تابع شباهت از دو تابع ریاضی اقلیدسی و منهتن استفاده شده است و تابع پاسخ ساختارهای نزدیک‌ترین، میانه، میانگین و معکوس را به خود خواهد گرفت. علاوه بر این تعداد سرویس‌های مشابه در بازه یک تا پنج در نظر گرفته شده‌اند.

الگوریتم تکامل تفاضلی: برای تمامی پایگاه داده‌ها اندازه جمعیت الگوریتم برابر ۳۰ و تعداد تکرارها ۱۰۰ گرفته شده است. ضریب همبری برابر ۰,۱ و ثابت تفاضلی ۲ می‌باشد؛ شمای انتخابی برای الگوریتم از نوع "DE/Best/1" در نظر گرفته شده.

الگوریتم ازدحام ذرات: با استفاده از مطالعات قبلی [۱۷] و روش آزمون و خطا، مقدار $C1$ و $C2$ برابر ۲ و وزن الگوریتم برابر ۱ گرفته شده است. اندازه جمعیت ۳۰ و تعداد تکرار ۱۰۰ می‌باشد. تابع برازش همان تابع در نظر گرفته شده برای الگوریتم تکامل تفاضلی است؛ در واقع هدف و معیار کارایی همه روش‌ها یکسان گرفته شده تا فرآیند مقایسه صحیح باشد.

الگوریتم ژنتیک: با استفاده از کارهای انجام شده قبلی [۱۷, ۲۸] و روش آزمون و خطا، ثابت همبری برابر ۰,۷ و ضریب جهش ۰,۳ در نظر گرفته شده است. در اینجا هم مثل دو روش قبل اندازه جمعیت ۳۰ و تعداد تکرار الگوریتم ۱۰۰ می‌باشد. تابع برازش همان تابع تعریف شده برای الگوریتم‌های تکامل تفاضلی و ازدحام ذرات می‌باشد. جدول ۳ تمامی پارامترهای موردنیاز برای هر یک از روش‌ها را به‌صورت خلاصه نشان می‌دهد.



شکل ۷- مقادیر پایگاه داده مصنوعی ناهمگن

جدول ۳- توصیف پارامترهای الگوریتم‌های مورد استفاده

الگوریتم	پارامترها و مقادیر
تکامل تفاضلی	ثابت همبری=۰,۱ ثابت جهش=۲ شما="DE/Best/1"
ازدحام ذرات	C1=C2=2 وزن=۱
ژنتیک	ثابت همبری=۰,۷ ثابت جهش=۰,۳ تعداد نخبه=۳
اندازه جمعیت=۳۰ ، تعداد تکرار=۱۰۰ تابع برازش=MMRE-PRED(0.25)	

۷- نتایج تجربی

پس از توضیح پایگاه داده‌ها و تنظیمات اولیه، در این بخش نتایج به دست آمده از روش‌های مختلف تخمین تلاش آورده شده است و نمودارها و جداول برای مقایسه بهتر رسم شده‌اند. در ابتدا مدل پیشنهادی با ساختارهای مختلف متد قیاس بر روی پایگاه ISBSG امتحان شده و نتایج به دست آمده در بخش بعد با دیگر مدل‌ها مقایسه شده‌اند. در آزمایش‌ها از چهار نسخه متد قیاس [۱۶] و چند روش مشهور تخمین تلاش [۹, ۱۰, ۱۷, ۲۴]، برای انجام مقایسه استفاده می‌شود. برای شبیه‌سازی تمامی روش‌ها از نرم‌افزار Matlab استفاده شده، چرا که اکثر توابع ریاضی و رگرسیون در جعبه ابزارهای[2] آن موجود است. علاوه بر این

[1] K Nearest Neighborhood (KNN)

[2] Toolbox

متد قیاس و ترکیب آن با سایر تکنیک‌ها به صورت جداگانه در این نرم‌افزار پیاده‌سازی شده‌اند.

جدول ۴- نتایج مدل پیشنهادی بر روی پایگاه داده ISBSG

تعداد سرویس	تابع شباهت	تابع پاسخ	MMRE	PRED
۱	اقلیدسی	نزدیک‌ترین	۱,۰۲۶۲	۰,۲۶۸۴
۱	منهتن	نزدیک‌ترین	۰,۹۷۶۶	۰,۲۶۳۱
۲	اقلیدسی	میانگین	۰,۹۰۴۶	۰,۲۷۰۲
		معکوس	۰,۹۳۵۷	۰,۲۷۰۲
۲	منهتن	میانگین	۰,۹۲۷۲	۰,۲۸۶۱
		معکوس	۰,۹۵۷۱	۰,۲۷۲۷
۳	اقلیدسی	معکوس	۰,۸۹۳۶	۰,۲۸۹۴
		میانگین	۰,۸۹۶۴	۰,۲۷۸۸
		میانه	۰,۷۲۲۸	۰,۲۸۵۹
۳	منهتن	معکوس	۰,۹۱۳۶	۰,۲۷۹۸
		میانگین	۰,۹۰۱۶	۰,۲۸۴۳
		میانه	۰,۷۱۶۶	۰,۲۹۹۰
۴	اقلیدسی	معکوس	۰,۹۷۳۰	۰,۳۰۳۰
		میانگین	۰,۸۴۳۲	۰,۲۸۰۶
		میانه	۰,۷۰۵۲	۰,۳۰۵۳
۴	منهتن	معکوس	۰,۸۷۳۶	۰,۲۹۲۹
		میانگین	۰,۸۹۱۵	۰,۲۹۴۷
		میانه	۰,۶۹۳۳	۰,۳۰۸۶
۵	اقلیدسی	معکوس	۰,۸۷۳۷	۰,۲۹۴۷
		میانگین	۰,۸۵۶۵	۰,۲۸۳۸
		میانه	۰,۶۵۰۴	۰,۳۱۶۷
۵	منهتن	معکوس	۰,۸۶۵۲	۰,۳۰۵۶
		میانگین	۰,۸۴۷۶	۰,۲۸۶۶
		میانه	۰,۶۴۰۹	۰,۳۴۱۷

۷-۱- نتایج بر روی پایگاه ISBSG

همان‌گونه که در بخش چهارم بیان شد متد قیاس، سه بخش حساس و قابل تنظیم دارد به نام‌های تابع شباهت، تابع پاسخ و تعداد سرویس‌های مشابه. در اینجا هر سه متغیر در مقایسه به کار گرفته شده و مقادیر متفاوت را دریافت کرده‌اند. مجموعه "یک تا پنج" برای تعداد سرویس مشابه، مجموعهٔ "اقلیدسی و منهتن" برای تابع شباهت و مجموعه "نزدیک‌ترین، میانه، میانگین و معکوس" برای تابع پاسخ در نظر گرفته شده‌اند. جدول ۴ نتایج به دست آمده از مدل پیشنهادی را بر پایه پارامترهای مختلف متد قیاس بر روی پایگاه ISBSG نشان می‌دهد. همان‌طور که از نتایج مشخص است پارامترهای مختلف تأثیر زیادی در نتایج به دست آمده داشته است.

با توجه به مقادیر، بهترین مورد برای تابع شباهت، فرمول منهتن و برای تابع پاسخ، حالت میانه می‌باشد. همچنین برای متغیر تعداد سرویس مشابه، مقدار ۱ بدترین و مقدار ۵ بهترین پاسخ را داشته‌اند. بنابراین مقادیر بزرگ‌تر متغیر تعداد سرویس مشابه و انتخاب سرویس‌های مشابه بیشتر برای عمل مقایسه، کارایی بهتری را از خود نشـان می‌دهد. همان‌گونه کـه مشـاهده می‌شود بین توابع اقلیدسی و منهتن تفاوت چندانی در نتیجه وجود ندارد؛ در مقابل در تمامی ترکیبات مقدار میانه برای تابع پاسخ بهترین جواب‌ها را داشته است. ضمناً برای تعداد سرویس مشابه برابر یک، هیچ انتخابی برای تابع شباهت وجود ندارد جز حالت نزدیک‌ترین (یعنی انتخاب تنها پاسخ موجود). در حالت تعداد سرویس مشابه برابر ۲، دو تابع پاسخ میانه و میانگین یک جواب را بر می‌گردانند که از نمایش حالت میانه در جدول خودداری شده است. لازم به ذکر است که بهترین ساختار برای مدل پیشنهادی بستگی به داده‌های مورد استفاده و خواص آنها دارد و برای داده‌های دیگر ممکن است پاسخ متفاوت شود. دقیق‌ترین پاسخ (MMRE=۰,۶۴۰۹ و PRED(0.25)=۰,۳۴۱۷) برای حالتی است که در آن تعداد سرویس مشابه، تابع شباهت و تابع پاسخ به ترتیب ۵، منهتن و میانه می‌باشند؛ این ساختار برای مقایسه با دیگر متدها و مدل‌های تخمین در بخش بعد استفاده خواهد شد.

۷-۲- مدل پیشنهادی در مقابل سایر مدل‌ها

در این بخش کارایی مدل پیشنهادی با ۹ مورد از بهترین مدل‌های تخمین تلاش مقایسه شده است. از آنجایی که یکی از اهداف این مقاله افزایش دقت روش قیاس بود، در تمامی مدل‌ها بهترین ساختار به دست آمده از اجرای روش قیاس برای مقایسه با دیگر مدل‌ها به کار گرفته شده است. جدول شماره ۵ خلاصه‌ای از بهترین نتایج به دست آمده بر روی پایگاه داده ISBSG را برای تمامی مدل‌های تخمین

نشان می‌دهد. برای نمایش بهتر، نتایج به‌صورت نمودار در شکل ۸ نشان داده شده‌اند. دو پارامتر کارایی MMRE و PRED(0.25) معرفی شده در بخش چهارم، برای مقایسه مدل‌ها استفاده شده است. از نتایج به دست آمده می‌توان مشاهده نمود که بهترین جواب متعلق به مدل پیشنهادی است (MMRE= ۰,۶۴۰۹ و PRED(0.25)=۰,۳۴۱۷) و بدترین پاسخ را درخت تصمیم می‌دهد (MMRE= ۱,۳۹۰۹ و PRED(0.25)=۰,۲۸۷۹). بهترین ساختار متد قیاس برای دو الگوریتم ژنتیک و ازدحام ذرات همانند مدل پیشنهادی، تابع شباهت منهتن و تابع پاسخ میانه می‌باشد که دلیل آن ویژگی‌های خاص پایگاه داده مورد استفاده می‌باشد. نتایج مدل‌های مختلف بر روی دیتاست مصنوعی همگن در جدول ۶ نشان داده شده است؛ شکل ۹ نیز نمای گرافیکی این نتایج را نشان می‌دهد. همان‌طور که از جدول مشخص است در بین همه روش‌ها مدل پیشنهادی بهترین کارایی را دارد (MMRE= ۰,۰۵۹۵ و PRED(0.25)=۰,۹۶۰) و در مقابل روش درخت تصمیم بدترین کارایی را از خود نشان داده است (MMRE= ۰,۰۸۵۶ و PRED(0.25)=۰,۹۲۳۳). مدل‌های ازدحام ذرات و ژنتیک به ترتیب پس از مدل پیشنهادی، بهترین کارایی را دارند. بنابراین تمامی روش‌های جستجو بر کارایی متد قیاس تأثیر مثبت می‌گذارند و آن را بهبود می‌دهند. همچنین دقت نتایج با توجه به ریاضی و مصنوعی بودن پایگاه داده، خیلی دقیق‌تر از پایگاه داده واقعی ISBSG می‌باشد. جدول ۷ مقادیر به دست آمده از روش‌های مختلف را بر روی پایگاه داده مصنوعی ناهمگن و شکل ۱۰ نمودار تصویری آن را نشان می‌دهد. در اینجا هم بهترین جواب متعلق به مدل پیشنهادی است (MMRE= ۰,۲۱۲۹ و PRED(0.25)=۰,۹۱۲۵) و بدترین کارایی را روش رگرسیون پله‌ای دارد (MMRE= ۰,۵۷۳۴ و PRED(0.25)=۰,۰۷۹۰). همچنین نزدیک‌ترین کارایی به مدل پیشنهادی را به ترتیب مدل‌های ژنتیک و ازدحام ذرات به دست آوردند. محدوده بالای تغییرات در این جدول نشان از پیچیدگی و تنوع بالای پایگاه داده مورد استفاده دارد. در این حالت نتایج نسبت به حالت واقعی ISBSG دقیق‌تر و نسبت به حالت مصنوعی همگن، بدتر شده است؛ چرا که تنوع و ناهمگنی داده‌ها در اینجا بیشتر است و در نتیجه مدل برای تخمین تلاش مشکل بیشتری دارد.

جدول ۵- نتایج مدل‌های مختلف تخمین روی پایگاه ISBSG

مدل	توضیح	MMRE	PRED
رگرسیون چندگانه	-	۰,۹۱۷۹	۰,۲۸۷۹
رگرسیون پله‌ای	-	۰,۶۷۸۴	۰,۲۴۲۴
درخت تصمیم	دسته‌بندی و رگرسیون	۱,۳۹۰۹	۰,۲۸۷۹
قیاس ۱	اقلیدسی و میانگین	۱,۰۲۲۷	۰,۳۱۸۹
قیاس ۲	اقلیدسی و معکوس	۰,۹۱۸۹	۰,۳۰۳۰
قیاس ۳	منهتن و میانه	۰,۸۰۵۱	۰,۳۳۳۳
قیاس ۴	منهتن و معکوس	۰,۹۳۶۰	۰,۲۸۷۹
قیاس ترکیبی	ژنتیک	۰,۶۸۶۳	۰,۲۶۷۷
قیاس ترکیبی	ازدحام ذرات	۰,۶۵۸۷	۰,۳۰۸۶
قیاس ترکیبی	تکامل تفاضلی	۰,۶۴۰۹	۰,۳۴۱۷

جدول ۶- نتایج مدل‌های مختلف روی پایگاه مصنوعی همگن

مدل	توضیح	MMRE	PRED
رگرسیون چندگانه	-	۰,۰۸۲۵	۰,۹۵۶۷
رگرسیون پله‌ای	-	۰,۰۶۷۲	۰,۹۰۳۲
درخت تصمیم	دسته‌بندی و رگرسیون	۰,۰۸۵۶	۰,۹۲۳۳
قیاس ۱	اقلیدسی و میانگین	۰,۰۷۸۵	۰,۹۳۶۷
قیاس ۲	اقلیدسی و معکوس	۰,۰۸۲۳	۰,۹۰۶۷
قیاس ۳	منهتن و میانه	۰,۰۶۰۴	۰,۹۵۰۰
قیاس ۴	منهتن و معکوس	۰,۰۷۹۸	۰,۹۰۶۷
قیاس ترکیبی	ژنتیک	۰,۰۶۵۰	۰,۹۵۶۷
قیاس ترکیبی	ازدحام ذرات	۰,۰۶۱۱	۰,۹۵۵۰
قیاس ترکیبی	تکامل تفاضلی	۰,۰۵۹۵	۰,۹۶۶۰

۷-۳- مقایسه زمان اجرای مدل‌ها

یکی از پارامترهای مورد مقایسه در مقالات علمی و بحث الگوریتم‌ها، زمان اجرای[1] آنها است. با وجود اینکه در حوزهٔ تخمین تلاش، دقت برآوردها بسیار مهم‌تر از زمان اجرا می‌باشد در این بخش به‌صورت جداگانه به بررسی پارامتر زمان می‌پردازیم. منظور از زمان اجرا مدت زمانی است که طی آن الگوریتم فرآیند برآورد متغیر را انجام می‌دهد. همان‌گونه که مشخص است عوامل متعددی بر مقدار زمان اجرا تأثیر می‌گذارند که از آن جمله می‌توان به مشخصات سیستم مورد استفاده، تنظیمات اولیه الگوریتم، نوع روش اعتبارسنجی و اندازهٔ ورودی اشاره کرد. در اینجا مشخصات سکوی اجرا عبارت است از: پردازندهٔ ۲ گیگا هرتز دو هسته‌ای، حافظهٔ اصلی[2] ۲ گیگابایتی و سیستم‌عامل ۳۲ بیتی ویندوز ۷. در همین راستا، تنظیمات اولیهٔ مدل‌ها و نوع اجرا مطابق با اطلاعات موجود در بخش ۶-۲ در نظر گرفته است؛ همچنین با توجه به اینکه ساختارهای مختلف روش قیاس تفاوتی در زمان اجرا نداشتند، از تکرار آنها در مقایسهٔ نتایج صرف‌نظر شده است. جدول ۸ نتایج به دست آمده از انجام شبیه‌سازی را بر روی هر سه مخزن داده نشان می‌دهد. همان‌گونه که از قبل نیز قابل پیش‌بینی بود مدل‌های ترکیبی، زمان اجرای بالاتری دارند. دلیل این افزایش را باید در ماهیت تکاملی این روش‌ها دنبال کرد؛ زیرا که مدل‌های ترکیبی از مفهوم تکرار[3] و اجرای چندباره استفاده می‌کنند. در مقابل، مرحلهٔ آموزش در مدل‌های واحد مثل رگرسیون‌ها و درخت تصمیم تنها یک‌بار طی می‌شود که خود آن نیز با توجه به ساختار ریاضی این مدل‌ها سربار زیادی ندارد. توجه به این نکته ضروری است که دقت برآوردها هدف اصلی و عمدهٔ پژوهشگران حوزهٔ تخمین تلاش را تشکیل می‌دهد و زمان اجرا چندان مطرح نیست؛ ضمن اینکه در تمامی فرآیند طولانی توسعهٔ سرویس، تخمین تلاش تنها یک بار و آن هم در فاز تحلیل انجام می‌شود. بنابراین بالا بودن زمان اجرای مدل پیشنهادی چندان نگران‌کننده و تأثیرگذار نخواهد بود.

جدول ۷- نتایج مدل‌ها روی پایگاه داده مصنوعی ناهمگن

مدل	توضیح	MMRE	PRED
رگرسیون چندگانه	-	۰,۲۹۱۴	۰,۸۲۶۷
رگرسیون پله‌ای	-	۰,۵۷۳۴	۰,۰۷۹۰
درخت تصمیم	دسته‌بندی و رگرسیون	۰,۳۵۴۶	۰,۸۰۳۳
قیاس ۱	اقلیدسی و میانگین	۰,۳۶۰۵	۰,۸۵۶۷
قیاس ۲	اقلیدسی و معکوس	۰,۳۳۱۵	۰,۸۳۶۷
قیاس ۳	منهتن و میانه	۰,۲۳۲۱	۰,۹۰۶۷
قیاس ۴	منهتن و معکوس	۰,۳۳۸۷	۰,۸۴۳۳
قیاس ترکیبی	ژنتیک	۰,۲۲۲۶	۰,۹۰۹۷
قیاس ترکیبی	ازدحام ذرات	۰,۲۲۳۹	۰,۹۰۸۹
قیاس ترکیبی	تکامل تفاضلی	۰,۲۱۲۹	۰,۹۱۲۵

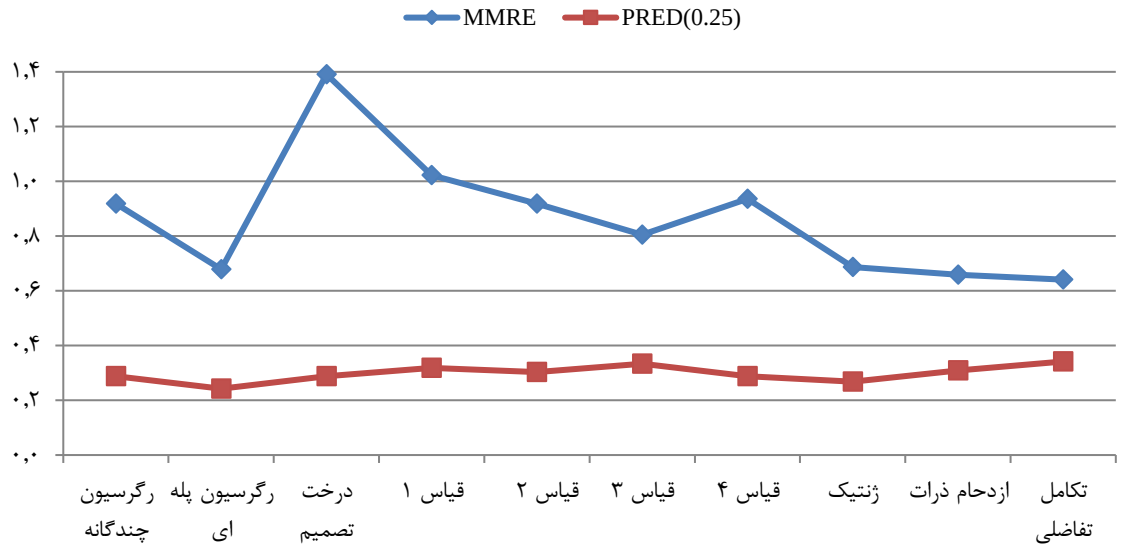
جدول ۸- مقایسه زمان اجرای مدل‌ها (واحد ثانیه)

مدل	توضیح	ISBSG	همگن	ناهمگن
رگرسیون چندگانه	-	۰,۰۲	۰,۰۰۱	۰,۰۰۱
رگرسیون پله‌ای	-	۰,۰۵	۰,۰۰۷	۰,۰۰۶
درخت تصمیم	دسته بندی و رگرسیون	۰,۱	۰,۰۱۵	۰,۰۱۴
قیاس معمولی	-	۱,۵	۰,۱۹	۰,۲۱
قیاس ترکیبی	الگوریتم ژنتیک	۵,۴۸	۶۷	۶۶
قیاس ترکیبی	الگوریتم ازدحام ذرات	۵,۲۶	۶۲	۶۰
قیاس ترکیبی	الگوریتم تکامل تفاضلی	۳,۹۵	۷۰	۶۹

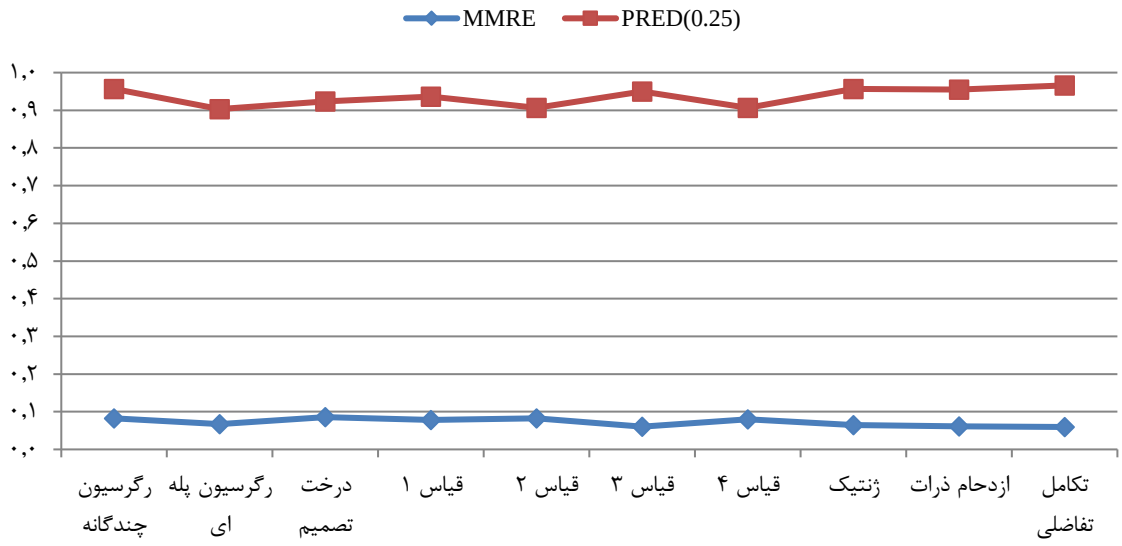
از مقادیر به دست آمده در جدول ۸ سه نکتهٔ مهم را می‌توان استخراج کرد. نخست آنکه ماهیت سرویس‌ها بر خلاف تعداد آنها، تأثیر چندانی بر زمان اجرا ندارد؛ زیرا که تغییر مخزن دادهٔ همگن به ناهمگن، زمان اجرا را تغییر نمی‌دهد. نکتهٔ دوم آنکه دقت یک مدل با زمان اجرای آن رابطهٔ معکوس دارد و با افزایش یکی دیگری کاهش می‌یابد.

[1] Run time
[2] RAM
[3] Iteration

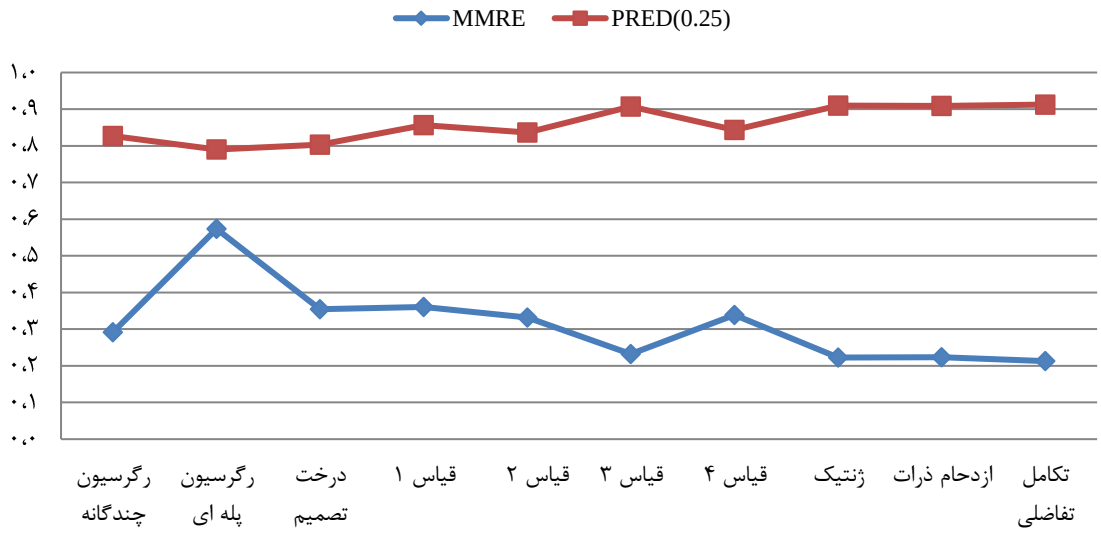
این مسئله نیز کاملاً منطقی به نظر می‌رسد و با مقایسهٔ اعداد جدول ۸ با جداول ۵، ۶ و ۷ قابل اثبات است. در نهایت، نکته سوم به این موضوع اشاره دارد که اندازهٔ مخزن داده به شدت روش‌های ترکیبی (الگوریتم‌های تکاملی) را تحت تأثیر قرار می‌دهد؛ در صورتی که مدل‌های واحد (مثل رگرسیون و درخت تصمیم) بیشتر از تعداد ویژگی‌های مخزن پیروی می‌کنند تا تعداد موجودیت‌های آن.



شکل ۸- نتایج مدل‌های مختلف تخمین تلاش بر روی پایگاه داده ISBSG



شکل ۹- نتایج مدل‌های مختلف تخمین تلاش بر روی پایگاه داده مصنوعی همگن



شکل ۱۰- نتایج مدل‌های مختلف تخمین تلاش بر روی پایگاه داده مصنوعی ناهمگن

۸- نتیجه‌گیری و کار آینده

تخمین تلاش لازم جهت توسعه سرویس‌های نرم‌افزاری نقش حیاتی در مدیریت سرویس بازی می‌کند. با توجه به تنوع بالا، پیچیدگی و غیر نرمال بودن سرویس‌های نرم‌افزاری تخمین دقیق تلاش، یک مسئله چالش برانگیز است. در دهه‌های اخیر انواع مدل‌ها و روش‌های تخمین در دو گروه الگوریتمی و غیر الگوریتمی ارائه شده‌اند که در این بین روش قیاس که از مقایسه یک سرویس با موارد مشابه قبلی استفاده می‌کند از مقبولیت بیشتری برخوردار است. در این مقاله یک روش جدید برای تخمین تلاش لازم جهت توسعه سرویس‌های نرم‌افزاری به کمک ترکیب دو متد قیاس و تکامل تفاضلی معرفی شد. مدل پیشنهادی با استفاده از دو مرحله آموزش و آزمون ساخته شده و ارزیابی آن بر روی پایگاه داده واقعی ISBSG و دو پایگاه داده مصنوعی متفاوت انجام گرفت. نتایج نشان از بهبود ملموس دقت تخمین و کاهش خطای آن داشت و هر دو پارامتر ارزیابی MMRE و PRED(0.25) به سمت مقادیر بهتر گرایش پیدا کردند. با توجه به جدید بودن حوزه سرویس‌های نرم‌افزاری و تخمین تلاش توسعه یک سرویس، مدل پیشنهادی قابل توسعه و کاربرد در موارد گسترده‌تر است. به عنوان مثال می‌توان دسته بندی دیگری از سرویس‌ها بر اساس نوع آنها ایجاد کرد و بجای یک پایگاه داده از چند پایگاه مجزا استفاده نمود و روند مقایسه را محلی و خاص کرد. انتخاب معیارهای کارایی متفاوت و ایجاد توابع برازش جدید از دیگر کارهای آینده است.

۹- مراجع

[1] Jorgensen, M., Shepperd, M. (2007). "A systematic review of software development cost estimation studies". IEEE Transactions on Software Engineering, Vol. 33, No. 1, pp. 33-53.

[2] Bardsiri, V.K., et al. (2013). "LMES: A localized multi-estimator model to estimate software development effort". Engineering Applications of Artificial Intelligence, Vol. 26, No. 10, pp. 2624-2640.

[3] Limam, N., Boutaba, R. (2010). "Assessing software service quality and trustworthiness at selection time". IEEE Transactions on Software Engineering, Vol. 36, No. 4, pp. 559-574.

[4] Boehm, B.W., R. Madachy, and B. Steece (2000). "Software cost estimation with Cocomo II with Cdrom". Prentice Hall PTR.

[5] Boehm, B.W. and R. Valerdi (2008). "Achievements and challenges in cocomo-based software resource estimation". Software, IEEE, Vol. 25, No. 5, pp. 74-83.

[6] Dalkey, N. and O. Helmer (1963). "An experimental application of the Delphi method to the use of experts". Management science, Vol. 9, No. 3, pp. 458-467.

[7] Azzeh, M. (2012). "A replicated assessment and comparison of adaptation techniques for analogy-based effort estimation". Empirical Software Engineering, Vol. 17, No. 1-2, pp. 90-127.

[8] Azzeh, M., Y. Elsheikh, and M. Alseid (2014). "An Optimized Analogy-Based Project Effort Estimation". International Journal of Advanced Computer Science & Applications, Vol. 5, No. 4, pp. 6-11.

[9] Breiman, L., et al. (1984). "Classification and regression trees". CRC press.

[10] Dejaeger, K., et al. (2012). "Data mining techniques for software effort estimation: A comparative study". IEEE Transactions on Software Engineering, Vol. 38, No. 2, pp. 375-397.

[11] Bardsiri, A.K. and S.M. Hashemi (2014). "Software Effort Estimation: A Survey of Well-known Approaches". International Journal of Computer Science Engineering, Vol. 3, No. 1, pp. 46-50.

[12] Jones, C. (2007). "Estimating software costs: bringing realism to estimating". McGraw-Hill Professional.

[13] Boehm, B.W. (1981). "Software engineering economics". Prentice-hall Englewood Cliffs (NJ).

[14] Albrecht, A.J. and Gaffney, J.E. (1983). "Software function, source lines of code, and development effort prediction: a software science validation". IEEE Transactions on Software Engineering, Vol. 9, No. 6, pp. 639-648.

[15] Andreou, A.S. and Papatheocharous, E. (2008). "Software Cost Estimation using Fuzzy Decision Trees". 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE '08). IEEE Computer Society, USA, pp. 371-374.

[16] Shepperd, M. and C. Schofield (1997). "Estimating software project effort using analogies". IEEE Transactions on Software Engineering, Vol. 23, No. 11, pp. 736-743.

[17] Bardsiri, V.K., et al. (2013). "A PSO-based model to increase the accuracy of software development effort estimation". Software Quality Journal, Vol. 21, No. 3, pp. 501-526.

[18] Azzeh, M. (2011). "Software effort estimation based on optimized model tree". 7th International Conference on Predictive Models in Software Engineering, Canada, ACM, pp. 1-8.

[19] Shukla, R., M. Shukla, and T. Marwala (2014). "Neural network and statistical modeling of software development effort". Second International Conference on Soft Computing for Problem Solving (SocProS), Springer, pp. 189-198.

[20] Azzeh, M., D. Neagu, and P.I. Cowling (2010). "Fuzzy grey relational analysis for software effort estimation". Empirical Software Engineering, Vol. 15, No. 1, pp. 60-90.

[21] Angelis, L. and I. Stamelos (2000). "A simulation tool for efficient analogy based cost estimation". Empirical software engineering, Vol. 5, No. 1, pp. 35-68.

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

[23] Hsu, C.-J. and C.-Y. Huang (2011). "Comparison of weighted grey relational analysis for software effort estimation". Software Quality Journal, Vol. 19, No. 1, pp. 165-200.

[24] Li, Y., M. Xie, and T. Goh (2007). "A study of genetic algorithm for project selection for analogy based software cost estimation". International Conference on Industrial Engineering and Engineering Management, Singapore, IEEE, pp. 1256-1260.

[25] Li, Y.-F., M. Xie, and T. Goh (2009). “A study of the non-linear adjustment for analogy based software cost estimation”. Empirical Software Engineering, Vol. 14, No. 6, pp. 603-643.
[26] Walkerden, F. and R. Jeffery (1999). “An empirical study of analogy-based software effort estimation”. Empirical software engineering, Vol. 4, No. 2, pp. 135-158.
[27] Wu, D., J. Li, and Y. Liang (2013). “Linear combination of multiple case-based reasoning with optimized weight for software effort estimation”. The Journal of Supercomputing, Vol. 64, No. 3, pp. 898-918.
[28] Ferrucci, F., et al. (2010). “Genetic programming for effort estimation: an analysis of the impact of different fitness functions”. Second International Symposium on Search Based Software Engineering (SSBSE), Italy, IEEE, pp. 89-98.
[29] Huang, S.-J. and N.-H. Chiu (2006). “Optimization of analogy weights by genetic algorithm for software effort estimation”. Information and software technology, Vol. 48, No. 11, pp. 1034-1045.
[30] Li, J., et al. (2007). “A flexible method for software effort estimation by analogy”. Empirical Software Engineering, Vol. 12, No. 1, pp. 65-106.
[31] Storn, R. and K. Price (1997). “Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces”. Journal of global optimization, Vol. 11, No. 4, pp. 341-359.
[32] Mohanty, B., S. Panda, and P. Hota (2014). “Controller parameters tuning of differential evolution algorithm and its application to load frequency control of multi-source power system”. International Journal of Electrical Power & Energy Systems, Vol. 54, pp. 77-85.
[33] Hu, Z., et al. (2014). “A convergent differential evolution algorithm with hidden adaptation selection for engineering optimization”. Mathematical Problems in Engineering, Vol. 2014, pp. 1-18.
[34] Nassif, A.B., D. Ho, and L.F. Capretz (2013). “Towards an early software estimation using log-linear regression and a multilayer perceptron model”. Journal of Systems and Software, Vol. 86, No. 1, pp. 144-160.
[35] Kocaguneli, E. and T. Menzies (2013). “Software effort models should be assessed via leave-one-out validation”. Journal of Systems and Software, Vol. 86, No. 7, pp. 1879-1890.
[36] Mair, C., M. Shepperd, and M. Jørgensen (2005). “An analysis of data sets used to train and validate cost prediction systems”. ACM SIGSOFT Software Engineering Notes, ACMpp, pp. 1-6.
[37] ISBSG (2011). “International Software Benchmarking standard Group”, www.isbsg.org.
[38] Pickard, L., B. Kitchenham, and S. Linkman (2001). “Using simulated data sets to compare data analysis techniques used for software cost modelling”. IEEE Proceedings-Software, Vol. 148, No. 6, pp. 165-174.
[39] Ahmed, M.A., M. Omolade Saliu, and J. AlGhamdi (2005). “Adaptive fuzzy logic-based framework for software development effort prediction”. Information and Software Technology, Vol. 47, No. 1, pp. 31-48.
[40] Li, Y.-F., M. Xie, and T.N. Goh (2009). “A study of project selection and feature weighting for analogy based software cost estimation”. Journal of Systems and Software, Vol. 82, No. 2, pp. 241-252.