

رویکردی جدید مبتنی بر سنجه‌های نرم‌افزاری جهت افزایش سودمندی آزمون بازگشت

مجتبی وحیدی اصل^۱، استادیار؛ محمدرضا دهقانی تفتی^۲، کارشناسی ارشد؛ علیرضا خلیلیان^۳، دانشجوی دکتری

۱- دانشکده مهندسی و علوم کامپیوتر - دانشگاه شهید بهشتی-تهران- ایران - mo_vahidi@sbu.ac.ir

۲- دانشکده مهندسی و علوم کامپیوتر - دانشگاه شهید بهشتی-تهران- ایران - mo.dehghani@mail.sbu.ac.ir

۳- گروه مهندسی نرم‌افزار - دانشکده مهندسی کامپیوتر- دانشگاه اصفهان- اصفهان- ایران - khalilian@eng.ui.ac.ir

چکیده: اولویت‌دهی آزمایش فنی است که اغلب برای کاهش هزینه‌های آزمون بازگشت نرم‌افزار استفاده شده‌است. فنون فعلی سعی کرده‌اند با کمک اطلاعات مختلف پوشش کد، قدرت آشکارسازی خطای هر آزمایش را تخمین بزنند و سپس با روشی ابتکاری آن‌ها را رتبه‌بندی نمایند. اما مطالعه‌ها نشان داده‌اند که پوشش هم‌بستگی قوی با سودمندی آزمایش‌ها و قدرت آن‌ها در آشکارسازی خطا ندارد. با تکیه بر مطالعه‌هایی که اثربخشی سنجه‌های کد را در پیش‌بینی خطاها نشان داده‌اند، حدس زده شد که می‌توان از اطلاعات حاصل از سنجه‌های کد برای طراحی فن مؤثری جهت اولویت‌دهی آزمایش‌ها بهره‌برداری نمود. بر مبنای این فرضیه، در این مقاله فن جدیدی برای اولویت‌دهی پیشنهاد می‌شود که براساس امتزاج داده روی اطلاعات سنجه‌های پیچیدگی کد کار می‌کند. نوآوری این تحقیق این است که قدرت آشکارکنندگی خطای آزمایش‌ها را در اولویت‌دهی با نگاه جدیدی تخمین می‌زند. برای ارزیابی فن پیشنهادی، آزمایش‌هایی روی نسخه‌های خطا دار هفت برنامه محک جاوا انجام داده شد. در آزمایش‌ها کارایی اولویت‌دهی اغلب حداقل ۷۰٪ بر حسب متوسط درصد آشکارسازی خطا مشاهده شد که این نتیجه فرضیه ما را معتبر می‌نماید.

واژه‌های کلیدی: آزمون نرم‌افزار، آزمون بازگشت، اولویت‌دهی آزمایش، سنجه‌های کد.

A New Approach Based on Software Metrics to Improve the Effectiveness of Regression Testing

M. Vahidi-Asl¹, Assistant Professor; M. R. Dehghani-Tafti², MSc; A. Khalilian³, PhD Student

1- Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran, Email: mo_vahidi@sbu.ac.ir

2- Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran, Email: mo.dehghani@mail.sbu.ac.ir

3- Software Engineering Department, Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran, khalilian@eng.ui.ac.ir

Abstract: Test case prioritization has been often used to alleviate the costs associated with software regression testing. Current techniques have attempted to estimate the fault exposing potential of test cases using code coverage information and rank them using a heuristic. However, studies show that coverage does not strongly correlate with the effectiveness and fault exposing potential of test cases. Relying on the results of studies that demonstrated the effectiveness of code metrics in fault prediction, we speculate that code metric information can be leveraged to design a new effective technique for test case prioritization. Based on our hypothesis, in this paper, a new prioritization technique is proposed that works based on data fusion on code complexity metrics. The novelty of our technique lies in its original viewpoint to estimate fault exposing potential of test cases in prioritization. To evaluate the proposed technique, we have conducted experiments on faulty versions of seven Java benchmarks. In the experiments, we often observed at least 70% performance in prioritization measured in terms of average percentage of fault detection, which validates our hypothesis.

Keywords: Software testing, regression testing, test case prioritization, software metrics.

تاریخ ارسال مقاله: ۱۳۹۶/۱۱/۲۷

تاریخ اصلاح مقاله: ۱۳۹۷/۰۴/۲۳ و ۱۳۹۷/۰۵/۰۲

تاریخ پذیرش مقاله: ۱۳۹۷/۰۸/۱۳

نام نویسنده مسئول: مجتبی وحیدی اصل

نشانی نویسنده مسئول: ایران - تهران - دانشگاه شهید بهشتی - دانشکده مهندسی و علوم کامپیوتر

۱- مقدمه

بر مبنای این فرضیه، در این مقاله فن جدیدی برای اولویت‌دهی پیشنهاد می‌شود. این فن با استفاده از امتزاج داده^۵ [۵] روی اطلاعات سنج‌های پیچیدگی کد طراحی شده است. هدف ما این است که فن اولویت‌دهی طراحی شود که با کمک اطلاعات حاصل از سنج‌های پیچیدگی و کیفیت کد، تخمین بهتری از قدرت آشکارکنندگی خطای آزمایش‌ها داشته باشد. نوآوری این تحقیق این است که قدرت آشکارکنندگی خطای آزمایش‌ها را در اولویت‌دهی با نگاه جدیدی تخمین می‌زند.

برای ارزیابی فن پیشنهادی، از هفت برنامه محک جاوا استفاده شده است. این برنامه‌ها از آخرین تحقیقات معتبر [۴] در آزمون بازگشت گرفته شده‌اند. فن پیشنهادی روی نسخه‌های خطا دار برنامه‌های محک اجرا شده و کارایی اولویت‌دهی براساس معیار متوسط درصد آشکارسازی خطا (APFD^۶) محاسبه شده است. این معیار به‌طور گسترده‌ای در ادبیات آزمون بازگشت مورد استفاده قرار گرفته است. در آزمایش‌ها کارایی اولویت‌دهی اغلب حداقل ۷۰٪ برحسب متوسط درصد آشکارسازی خطا مشاهده شد که این نتیجه فرضیه ما را معتبر می‌نماید.

به‌طور خلاصه، نوآوری‌های این مقاله عبارت‌اند از:

- پیشنهاد استفاده از سنج‌های پیچیدگی کد برای اولویت‌دهی آزمایش‌ها در آزمون بازگشت
- ترکیب مجموعه آزمون‌های اولویت‌دهی شده توسط سنج‌های برتر به‌وسیله یکی از روش‌های امتزاج داده
- آزمایش‌های تجربی برای اولویت‌دهی روی هفت برنامه محک جاوا و مقایسه با روش پوشش کامل
- بحثی در نتایج به‌دست‌آمده و عوامل زمینه‌ای در موفقیت روش پیشنهادی

ساختار ادامه مقاله به این شرح است: در بخش دوم مفاهیم لازم برای درک مقاله و بیان مسئله ارائه می‌شود. روش پیشنهادی در بخش سوم تشریح می‌گردد. بخش چهارم به شرح ارزیابی، آزمایش‌ها، نتایج به‌دست آمده و بحث در نتایج می‌پردازد. در بخش پنجم برخی از مهم‌ترین کارهای مرتبط مرور می‌شود و مقاله با نتیجه‌گیری در بخش ششم به پایان می‌رسد.

۲- مفاهیم لازم و بیان مسئله

آزمون بازگشت یکی از فعالیت‌های رایج برای تضمین کیفیت نرم‌افزار است. این آزمون پس از هر بار تغییر کد انجام می‌شود تا اطمینان حاصل شود تغییرات خطایی ایجاد نکرده باشند. همچنین بررسی می‌کند قسمت‌هایی از نرم‌افزار که در دست کار می‌کردند تحت تأثیر منفی قرار نگرفته باشند. برای این منظور آزمایش‌هایی روی برنامه اجرا می‌شود که آن‌ها را مجموعه آزمون بازگشت می‌نامند. به‌طور ساده، آزمایش عبارت است از مقادیر ورودی موردنیاز برای تکمیل اجرای نرم‌افزار تحت آزمون، خروجی‌های مورد انتظار و سازوکاری برای ارزیابی

آزمون بازگشت^۱ یکی از فعالیت‌های مهم در مرحله نگهداری نرم‌افزار است. هم‌زمان با بزرگ‌تر شدن پروژه‌ها و افزایش تعداد آزمایش‌ها، آزمون بازگشت پرهزینه‌تر شده و انجام آزمون با اجرای همه آزمایش‌ها هم غیرعملی می‌گردد [۱۴]. طبق گزارش‌ها [۱۷-۱۵]، آزمون بازگشت ۸۰٪ از بودجه آزمون نرم‌افزار را به خود اختصاص می‌دهد و اجرای برخی از مجموعه‌های آزمون بیش از هفت هفته به طول می‌انجامد. برای مثال، محصول نرم‌افزاری یک شرکت مجموعه آزمونی با بیش از ۳۰ هزار آزمایش داشته است. این آزمایش‌ها به بیش از ۱۰۰۰ ماشین-ساعت زمان برای اجرا و صدها ساعت برای نظارت مهندسان بر فرآیند آزمون بازگشت نیاز داشته‌اند [۱۸].

برای کاهش هزینه‌های آزمون بازگشت فنون متعددی پیشنهاد شده است. یکی از فنون اثربخش، اولویت‌دهی آزمایش^۲ است. اولویت‌دهی سعی می‌کند به شیوه‌ای قدرت آشکارکنندگی خطای آزمایش‌ها را تخمین بزند و آن‌ها را بر این اساس رتبه‌بندی نماید. هدف این است که آزمایش‌های مؤثرتر زودتر اجرا شوند و خطاها سریع‌تر آشکار گردند. شمار بسیاری از فنون موجود قدرت آشکارکنندگی خطا را برحسب معیارهای مختلف پوشش^۳ تخمین می‌زند. متأسفانه تکیه بر پوشش مشکلاتی دارد. اغلب خطاهای پنهان در نقاطی بسیار خاص از کد اتفاق می‌افتند که پوشش بالای کد آزمایش، لزوماً منجر به یافتن آن‌ها نمی‌شود. ضمن آن که اطلاعات پوشش اولیه یک آزمایش با حرکت به سوی نسخه‌های بعدی نرم‌افزار، کم‌اثرتر می‌شوند و درجایی ممکن است همراه‌کننده شوند. در نتیجه، بهبود آزمون بازگشت با در نظر گرفتن اطلاعات پوشش کد همیشه اثربخش نیست و نمی‌تواند به‌خوبی خطاهای ناشی از تغییرات را آشکار سازد [۱۹]. اختصاصاً نشان داده شده [۲۰] که پوشش همبستگی قوی با سودمندی مجموعه آزمون و قدرت آشکارکنندگی خطای آن ندارد.

مشکل دیگر این است که استفاده از اطلاعات پوشش از نسخه‌های قبلی، اثربخشی آزمون بازگشت برای نسخه‌های بعدی را کاهش می‌دهد. نشان داده شده است [۳] که افزایش فاصله بین نسخه‌های آزمون نرم‌افزار که برای جمع‌آوری اطلاعات پوشش استفاده می‌شود باعث کاهش قابل‌ملاحظه سودمندی فنون اولویت‌دهی می‌گردد. نتایج تحقیق لو^۵ و همکاران [۳]، اهمیت استفاده از اطلاعات پوشش به‌روز و جدید برای آزمون بازگشت را نشان می‌دهد.

انگیزه تحقیق حاضر این است که چگونه می‌توان با در نظر گرفتن اطلاعات دیگر در کنار اطلاعات پوشش، اثربخشی فن اولویت‌دهی افزایش یابد. با تکیه بر مطالعه‌هایی [۲۴-۲۱] که اثربخشی سنج‌های کد را در پیش‌بینی خطاها و خرابی‌ها نشان داده‌اند، حدس این است که می‌توان روش ابتکاری طراحی کرد که با اطلاعات حاصل از سنج‌های کد، اولویت‌دهی مؤثری انجام دهد.

استفاده است. تصدیق می‌شود که روش پیشنهادی در این مقاله ارتباط معنایی بین معیارها و سنج‌های پیچیدگی ساختار یک برنامه (در هر بن‌انگاره^۸ برنامه‌سازی همچون شی‌گرایی) و آزمون بازگشت را مورد بررسی و ملاحظه قرار نداده است. در حقیقت روش پیشنهادی این مقاله اولین اکتشاف ما در این زمینه است. قطعاً مطالعه‌های نظری و تجربی بیش‌تر لازم است که با تعیین این ارتباط‌های معنایی، روابط ریاضی مؤثرتری برای بهره‌برداری و ترکیب سنج‌ها طراحی نماید.

چهارچوب روش پیشنهادی در شکل (۱) نشان داده شده است. مرحله اول روش پیشنهادی نیازمند جمع‌آوری اطلاعاتی از کد است. برای این کار آزمایش‌ها بر روی برنامه‌ها اجرا می‌شوند. خروجی این عمل، میزان پوشش عناصر مختلف به ازای آزمایش‌های برنامه می‌باشد؛ یعنی مشخص می‌شود که هر آزمایش چه عناصری از برنامه را پوشش داده است. سپس اطلاعات سنج‌های کد برنامه در سطح کلاس و متد استخراج می‌شوند. محدوده روش پیشنهادی برنامه‌های زبان‌های شی‌گرا است هرچند برای سایر زبان‌ها هم قابل استفاده است.

در ادامه این دو نوع اطلاع مبتنی بر پوشش و مبتنی بر سنج با یکدیگر ترکیب می‌شوند. به عبارتی مشخص می‌شود که آزمایش‌ها، چه عناصری از برنامه را پوشش داده‌اند و مقدار سنج کد برای آن عناصر پوشش داده شده چقدر است. سپس امتیاز هر آزمایش به ازای هر سنج قابل محاسبه است. به ازای هر سنج، جمع مقدار سنج عناصر پوشش داده شده توسط هر آزمایش، امتیاز آن آزمایش محسوب می‌شود. اکنون به ازای هر سنج، به آزمایش‌ها امتیازی داده شده است. پس از مرتب‌سازی امتیازهای داده‌شده، به ازای هر سنج، مجموعه آزمون مرتب شده‌ای وجود دارد. ارزیابی مجموعه آزمون‌های مرتب شده توسط سنج‌ها معمولاً توسط معیار APFD صورت می‌گیرد.

آزمون بازگشت به تغییرها در نسخه‌های مختلف یک نرم‌افزار نیاز دارد. هنگام ارزیابی یک فن اولویت‌دهی، این تغییرها در قالب نسخه‌های خطا در نظر گرفته می‌شوند. نسخه‌های خطا با کاشتن خطا توسط ابزارهای آزمون جهش در برنامه ساخته می‌شوند. کاشتن این خطاها باعث ایجاد نسخ تغییر یافته می‌شود. عمل کاشتن خطا و رسیدن به نسخ خطا در قسمت آزمون جهش در شکل (۱) انجام می‌شود. پس از ارزیابی، سنج‌هایی که از لحاظ APFD برتر بوده‌اند مشخص می‌شوند. سپس برای بهبود مقدار APFD به ازای یک برنامه، از مجموعه آزمون‌های سنج‌های برتر استفاده کرده و آن‌ها با روش ترکیب خطی ترکیب می‌شوند تا مجموعه آزمون مرتب‌شده جدید و بهبود یافته‌ای حاصل شود. در بخش‌های بعدی جزئیات بیش‌تری از اجزای روش پیشنهادی ارائه می‌شود.

۳-۱- جمع‌آوری اطلاعات پوشش آزمون

در این مرحله اطلاعات پوشش آزمایش‌های برنامه محاسبه می‌شود. این کار توسط ابزار جمع‌آوری پوشش کد انجام می‌گردد. این ابزار آزمایش‌ها را بر روی کد برنامه اجرا می‌کند و در خروجی، به ازای آزمایش‌های مختلف، میزان پوشش عناصر مختلف برنامه را مشخص می‌کند. بدین

خروجی برنامه تحت آزمون (اوراکل آزمون) [۳۴]. از آنجائی که آزمون بازگشت بارها تکرار می‌شود، هزینه اجرایی مجموعه آزمون بازگشت زیاد می‌شود. یکی از روش‌های کارا تر کردن آزمون بازگشت اولویت‌دهی آزمایش‌هاست؛ آزمایش‌ها طوری مرتب شوند که آن‌هایی که شانس آشکار سازی خطا دارند، زودتر اجرا شوند. تعریف اولویت‌دهی این است [۱۵]:

ورودی: یک مجموعه آزمون T ، مجموعه‌ای از جایگشت‌های T مو سوم به PT ، و تابعی از PT به اعداد حقیقی، $f = PT \rightarrow \mathbb{R}$.
مسئله: یافتن $T' \in PT$ به طوری که:
 $(\forall T'')(T'' \in PT)(T'' \neq T') [f(T') \geq f(T'')]$
 تابع f بیانگر کمیتی است که باید بیشینه گردد.

از آنجا که مسئله اولویت‌دهی، NP سخت است، از روش‌های ابتکاری برای حل آن استفاده می‌شود. برای اولویت‌دهی معمولاً اطلاعاتی از برنامه استخراج می‌شود تا به کمک آن‌ها قدرت آشکارکنندگی خطای هر آزمایش را تخمین بزنند. یکی از اطلاعات رایج، پوشش کد بوده است. پوشش کد معمولاً بر اساس معیارهای مختلفی مانند شرط صورت می‌گیرد. در این صورت، پوشش شرط آزمایش t یعنی دستورهای شرطی که پس از اجرای t روی برنامه اجرا می‌شوند.

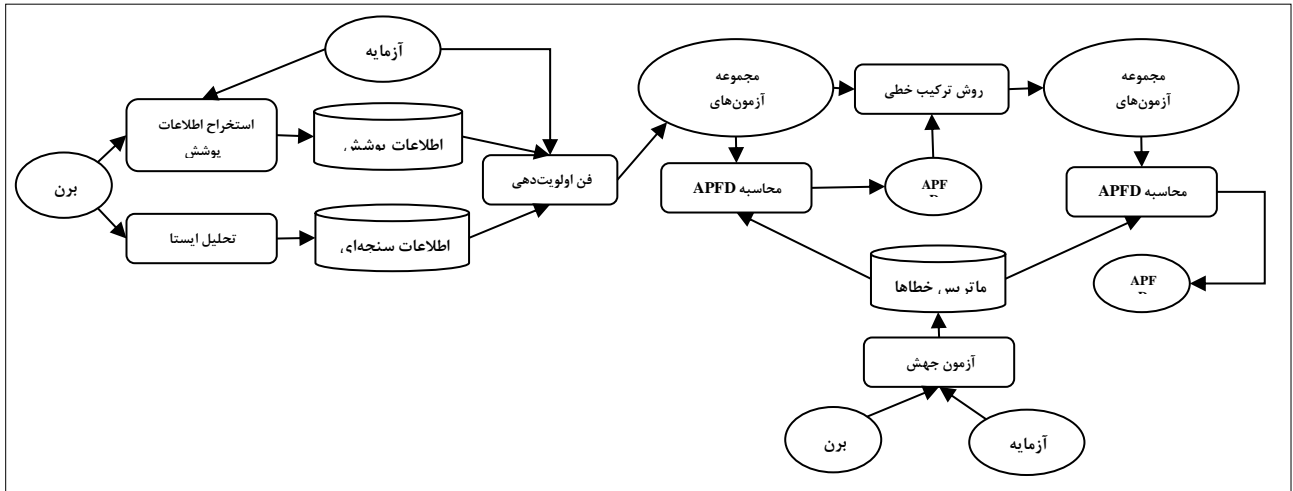
در مسئله اولویت‌دهی، f تعداد خطاهایی است که هرچه زودتر توسط آزمایش‌ها آشکار شده‌اند. برای تخمین f معمولاً از معیار APFD استفاده می‌شود. این معیار میانگین وزن‌داری از درصد خطاهایی که در طول عمر مجموعه آزمون کشف شده‌اند را اندازه‌گیری می‌کند. محدوده مقادیر APFD در بازه صفر تا ۱۰۰ قرار دارد. مقادیر بزرگ‌تر نشان‌دهنده نرخ تشخیص خطای بالاتر (بهتر) است. برای تعریف APFD به طور رسمی‌تر، فرض کنید T مجموعه آزمونی باشد که شامل n آزمایش است و F مجموعه‌ای از m خطا است که توسط T آشکار می‌شوند. فرض کنید TF_i مکان اولین آزمایش‌ای باشد که در ترتیب T' از T ، خطای i را آشکار می‌سازد. مقدار APFD برای مجموعه آزمون T' طبق رابطه (۱) محاسبه می‌شود [۱۵].

$$APFD = 1 - \frac{TF_1 + TF_2 + \dots + TF_m}{nm} + \frac{1}{2n} \quad (1)$$

۳- روش پیشنهادی

در این بخش فرضیه و رویکرد پیشنهادی ارائه می‌گردد. فرضیه اصلی این مقاله این است که تغییرات برنامه منجر به تغییر مقادیر سنج‌ها می‌شوند و چنین تغییراتی باعث می‌شود امتیاز و اولویت آزمایش‌ها نیز تغییر کند. در نتیجه، آزمون بازگشت به تغییرات حساسیت نشان می‌دهد. به عبارت دیگر، محل رخداد تغییرات باعث می‌شود مقادیر بعضی از سنج‌ها برای آزمایش‌های پوشاننده آن محل‌ها تغییر کند و این تغییر بر امتیاز و اولویت آن آزمایش‌ها اثر بگذارد. در نتیجه رویکرد پیشنهادی علاوه بر نسخه اصلی برنامه، روی نسخه‌های تغییر کرده برنامه قابل

معنا که به‌ازای هر آزمایش مشخص می‌شود که چه عناصری از برنامه را در سطوح مختلف کلاس و متد پوشش داده است.



شکل ۱: نمای کلی راه‌کار پیشنهادی

در انتخاب زیرمجموعه‌ای از سنج‌ها، تصمیم گرفته شد سنج‌هایی انتخاب شوند که در مطالعه‌های خطاپروری [۲۳، ۲۴] مورد استفاده قرار گرفته‌اند. درحقیقت این مقاله‌ها برای پیش‌بینی خطاها و خرابی‌های نرم‌افزار در زمینه مورد بررسی شان از این سنج‌ها استفاده کرده‌اند. تصدیق می‌شود که این تصمیم بسیار ابتدایی است و ممکن است بعضی از سنج‌ها با هم هم‌پوشانی داشته‌باشند یا بعضی از سنج‌های مهم‌تر نادیده گرفته شده باشند. بااین‌حال، هدف اصلی پیشنهاد این مقاله و آزمایش‌های آن، معتبر کردن این ایده است که سنج‌های کد در زمینه اولویت‌دهی آزمایش‌ها قابل استفاده‌اند. مطالعه‌های تکمیلی مورد نیاز است که سنج‌های بیشتری را مورد بررسی قرار دهد، همبستگی سنج‌ها را به‌دست آورد و زیرمجموعه مؤثرتری به‌خصوص برای هر دامنه برنامه‌سازی انتخاب نماید.

جدول ۱: سنج‌های پیچیدگی کد در سطح متد

نام	توضیح
(FANIN) CountInput	تعداد ورودی‌های یک متد
(FANOUT) CountOutput	تعداد خروجی‌های یک متد
(NPATH) CountPath	تعداد مسیرهای منحصره‌فرد داخل بدنه یک متد
CountPathLog	لگاریتم در مبنای ۱۰ سنج NPATH
CountStmt	تعداد دستورالعمل‌های اعلائی و اجرایی
Knots	تعداد تقاطع‌های جریان کنترلی
MinEssentialKnots	تعداد knot های کد کاهش یافته برنامه
MaxNesting	حداکثر تعداد سطوح تودرتویی ساختارهای کنترلی در یک متد
Cyclomatic Complexity	تعداد نقاط تصمیم در کد به‌علاوه یک
CyclomaticStrict	سنج قبلی به‌علاوه تعداد رخداد‌های and و or در کد
Essential Complexity	Cyclomatic complexity کد کاهش یافته برنامه

۳-۲- جمع‌آوری اطلاعات سنج‌های کد

شواهد موجود در مطالعه‌های قبل [۲۷-۲۲] نشان می‌دهد که می‌توان از سنج‌های کد برای پیش‌بینی خطاها و خرابی‌ها و نیز خطاگیری کد استفاده کرد. به‌عبارتی می‌توان اطلاعات مفیدی را از مسیر توسعه نرم‌افزار استخراج کرد و با استفاده از سنج‌های کد، اولویت‌دهی آزمایش‌ها را بهبود داد. مزیت این اطلاعات این است که بدون نیاز به اجرای آزمایش‌ها و فراکدگذاری^۱ به دست می‌آید. سنج‌ها رابطه مستقیم با تعداد خطاها در عناصر برنامه دارند. به‌عبارتی عنصری از برنامه که مقدار هر یک از سنج‌هایش از مقدار سنج متناظر عنصر دیگری از برنامه بیشتر باشد، احتمال خطاخیزبودن آن عنصر بیشتر است. با این اوصاف می‌توان از اطلاعات سنج‌ها برای اولویت‌دهی آزمایش‌ها استفاده کرد.

در این مرحله اطلاعات سنج‌های کد از برنامه استخراج می‌شود. برای این کار از ابزارهای تحلیل ایستا استفاده می‌کنیم که اطلاعات سنج‌های کد را به صورت ایستا از برنامه استخراج می‌کنند. ابزارهای مختلف برای استخراج این اطلاعات، کتابخانه‌ها و واسطه‌هایی را در اختیار برنامه‌نویسان قرار می‌دهند. بدین‌صورت که می‌توان اطلاعات سنج‌های مختلف عناصر کد را از طریق ارسال درخواست به آن ابزارها، به دست آورد.

سنج‌های کد را برای زبان‌های شی‌گرا می‌توان به دودسته سنج‌های سطح متد و سطح کلاس تقسیم‌بندی کرد [۲۳، ۲۴]. پس می‌توان به‌ازای هر برنامه دو نوع بردار در نظر گرفت. یک بردار بیان‌گر اطلاعات سنج‌های سطح کلاس برنامه و دیگری بیان‌گر اطلاعات سنج‌های سطح متد برنامه است. در این تحقیق از ۱۱ سنج در سطح متد و ۱۶ سنج در سطح کلاس استفاده می‌شود. این سنج‌ها به‌ترتیب در جدول‌های (۱) و (۲) خلاصه شده‌اند.

۳-۳- ترکیب اطلاعات پوشش و اطلاعات سنجه

در این مرحله اطلاعات جمع‌آوری شده در مرحله قبل با یکدیگر ترکیب می‌شوند، یعنی ترکیب ماتریس پوشش آزمون با بردار سنجه‌های کد. برای این کار باید سنجه‌هایی که در سطح کلاس هستند با ماتریس پوشش سطح کلاس ترکیب شوند و سنجه‌هایی که در سطح متد هستند با ماتریس پوشش سطح متد ترکیب شوند. این ماتریس نشان می‌دهد که هر آزمایه چه عناصری از برنامه را پوشش داده است و مقدار سنجه مورد نظر برای آن عنصر برنامه چقدر است.

جدول ۲: سنجه‌های پیچیدگی کد در سطح کلاس

نام	توضیح
PercentLackOfCohesion (LCOM/LOCM)	یک منهای (انسجام بین داده‌های یک کلاس و متدهای آن)
(DIT) MaxInheritanceTree	عمق یک کلاس در سلسله مراتب وراثت
(NOC) CountClassDerived	تعداد زیرکلاس‌های مستقیم یک کلاس
(CBO) CountClassCoupled	تعداد کلاس‌های وابسته به یک کلاس
(IFANIN) CountClassBase	تعداد کلاس‌های پدر مستقیم یک کلاس
CountDeclClassMethod	تعداد متدهای یک کلاس
(NV) CountDeclClassVariable	تعداد متغیرهای یک کلاس
AvgCyclomaticStrict	میانگین CyclomaticStrict از تمامی متدهای یک کلاس
(AMS) AvgLineCode	میانگین تعداد خطوط متدهای یک کلاس
(NM) CountDeclMethodAll	تعداد متدهای یک کلاس شامل متدهایی هم که به ارث برده
CountDeclMethodDefault	تعداد متدهای محلی عمومی
(NPRM) CountDeclMethodPrivate	تعداد متدهای خصوصی که به ارث برده نشده‌اند
MaxCyclomatic	بیشینه پیچیدگی سیکلوماتیک متدهای یک کلاس
MaxEssential	بیشینه Essential Complexity متدهای یک کلاس
(WCM) SumCyclomatic	مجموع پیچیدگی سیکلوماتیک تمامی متدهای کلاس
SumEssential	مجموع Essential Complexity متدهای یک کلاس

به‌عنوان نمونه، فرض کنید سه آزمایه مفروض است و برنامه ما دو کلاس و دو متد داشته باشد. آزمایه‌ها در سطرهای ماتریس و نام کلاس‌ها و متدها در ستون‌های ماتریس قرار می‌گیرند. در این صورت درایه سطر i و ستون j نشان می‌دهد آزمایه i آیا متد یا کلاس j را در اجرا پوشش داده است یا خیر (۱ یا ۰ به ترتیب). سپس به‌ازای درایه‌های ۱ (یعنی پوشش داده شده) مشخص می‌شود که مقدار انواع سنجه‌های کلاسی/متدی به‌ازای آن عنصر j و آزمایه i چقدر است.

فرض کنید آزمایه T_i عناصری از برنامه را پوشش می‌دهد که مقدار سنجه بیش‌تری نسبت به آزمایه T_j دارد که عناصر تحت پوشش آن، مقدار سنجه متناظر کمتری دارند. با توجه به ارتباط بین سنجه‌ها و خطاخیزی، آزمایه T_i پتانسیل بیشتری در تشخیص خطاها نسبت به

آزمایه T_j دارد. پس می‌توان از اطلاعات سنجه‌ها برای اولویت‌دهی آزمایه‌ها استفاده کرد.

برای این کار، به‌ازای هر آزمایه، مقدار سنجه‌های عناصر برنامه را در صورتی که آن عنصر توسط آن آزمایه پوشش داده شده باشد، با هم جمع می‌شوند. مقدار حاصل برابر است با امتیازی که آن سنجه به آن آزمایه داده است؛ به عبارتی این عدد قدرت تشخیص خطای آن آزمایه از دید آن سنجه تخمین می‌زند. اکنون می‌توان با توجه به امتیازی که هر سنجه برای هر آزمایه داده است، اولویت‌دهی را انجام داد.

۳-۴- ساخت ماتریس خطاها

برای ارزیابی مجموعه آزمون‌های مرتب شده توسط سنجه‌ها، خطاهای کاشته شده در برنامه لازم هستند. این خطاها توسط ابزارهای آزمون جهش به دست می‌آیند. ابزارهای آزمون جهش نیاز به برنامه و مجموعه آزمون آن دارند. این ابزارها این دو قلم اطلاعات را دریافت می‌کنند و با استفاده از عملگرهای جهشی اقدام به کاشتن خطا در برنامه می‌نمایند. یعنی با استفاده از این عملگرها، در برنامه تغییراتی ایجاد می‌شود و به‌عنوان خطا در نظر گرفته می‌شوند. سپس به‌ازای هر خطای کاشته‌شده، آزمایه‌ها اجرا می‌گردند و آن آزمایه‌هایی که قادر به تشخیص آن خطا باشند، مشخص می‌شوند. بنابراین با استفاده از این ابزار، خطاهای مختلفی در برنامه کاشته می‌شوند و به‌ازای هر خطای کاشته شده مشخص می‌شود که کدام آزمایه‌ها توانسته‌اند آن خطا را تشخیص دهند. در نتیجه ماتریسی حاصل می‌شود که نشان می‌دهد هر آزمایه کدام‌یک از خطاهای کاشته‌شده در برنامه را آشکار می‌کند.

۳-۵- ارزیابی مجموعه آزمون‌های اولویت‌دهی شده

برای ارزیابی مجموعه آزمون‌های مرتب‌شده توسط سنجه‌ها، از معیار APFD مشابه مقالات قبلی [۳، ۴] استفاده می‌شود. این تحقیقات در مرحله ارزیابی، برای شبیه‌سازی نسخه‌های بعدی خطادار، از خطاهای کاشته شده با استفاده از ابزار جهش استفاده کرده‌اند. بدین منظور، به تعداد خاصی در آخرین نسخه از نرم‌افزار، خطا کاشته می‌شود. سپس این خطاهای کاشته‌شده در تعداد دسته‌های ثابتی به نام گروه خطا تقسیم‌بندی می‌شوند. پس هر گروه خطا شامل تعداد خاصی خطای کاشته شده است. همچنین تضمین می‌شود که خطاهای انتخابی می‌توانند حداقل توسط یک آزمایه تشخیص داده شوند و گروه‌های مختلف خطا، خطاهای یکسان ندارند. در نتیجه به تعداد گروه‌های خطا، نسخه‌های خطادار موجودند. حال توانایی تشخیص خطای مجموعه آزمون‌های مرتب‌شده به‌ازای سنجه‌ها با استفاده از معیار APFD بر روی این نسخه‌های خطادار، اندازه‌گیری می‌شوند.

همچنین برای ارزیابی مقایسه‌کردن مجموعه آزمون‌های تولید شده با فنون کنترلی (مبنای مقایسه) لازم می‌شود. در این مقاله اولویت‌دهی پوشش کامل به‌عنوان فن کنترلی در نظر گرفته می‌شود [۱، ۲]. اولویت‌دهی پوشش کامل به آزمایه‌ای که متدهای بیشتری از برنامه را پوشش دهد، اولویت بیشتری تخصیص می‌دهد. گاهی فنون نتایج خود

۳-۶- ترکیب سنج‌ها

اکنون از بین سنج‌هایی که در قسمت قبل نسبت به فن کنترلی نتیجه بهتری داشته‌اند، k تا از بهترین‌هایشان انتخاب می‌شوند. ایده اصلی این است که هر کدام از سنج‌ها از جنبه‌ای به عناصر ساختاری برنامه (کلاس یا متد) نگاه می‌کنند و از آن جنبه به لحاظ خطاخیزی به آن‌ها امتیاز می‌دهند. با استفاده از یکی از فنون امتزاج داده، اطلاعات این سنج‌های برتر ترکیب می‌شوند تا اولویت‌دهی آزمایش‌ها به صورت بهتری انجام گیرد. در سال‌های اخیر فنون امتزاج داده در حوزه‌های مختلفی برای بهبود اثربخشی استفاده شده‌اند [۵].

در این تحقیق از روش ترکیب خطی که یکی از روش‌های امتزاج داده است، استفاده شده است [۵]. در واقع هدف محاسبه اولویت هر آزمایش است با توجه به اولویت‌دهی‌هایی که سنج‌های برتر به آزمایش‌ها داده‌اند. برای این کار باید میزان اهمیت هر سنج را بدانیم. بنابراین هدف، محاسبه میزان اهمیت هر سنج در امتیازی که به آزمایش‌ها می‌دهد، می‌باشد. در این روش امتیاز هر آزمایش به صورت رابطه (۲) محاسبه می‌شود.

$$Score(tc_i) = \sum_{j=1}^k w_j \times s_{ij} \quad (2)$$

که در رابطه بالا، tc_i آزمایش i ام است. مقدار k برابر تعداد سنج‌های برتر می‌باشد. w_j برابر وزن تخصیص یافته به سنج j ام با استفاده از روش ترکیب خطی است. سادگی و هزینه کم محاسباتی یکی از دلایل انتخاب ترکیب خطی است. به علاوه چون در این تحقیق میزان اهمیت و تأثیر سنج‌ها مورد بررسی قرار گرفته است، استفاده از ترکیب خطی اولین انتخاب منطقی به نظر می‌رسد. s_{ij} نیز برابر با مقدار نرمال شده سنج j ام برای آزمایش i ام است. نرمال‌سازی محدوده مقادیر را در بازه بین صفر و یک قرار می‌دهد. اکنون w_j ها مجهول هستند.

برای یافتن w_j ها لازم است معیاری به نام همپوشانی بین دو مجموعه مرتب‌شده معرفی شود. میزان همپوشانی دو مجموعه مرتب‌شده می‌تواند معیاری از شباهت یا هم‌بستگی آن دو مجموعه باشد. یکی از روش‌ها برای یافتن میزان همپوشانی بین دو مجموعه که توسط وو [۵] معرفی شده است، به صورت رابطه (۳) می‌باشد.

$$o_rate_{ij} = \frac{|L_i \cap L_j|}{n} \quad (3)$$

در این رابطه o_rate_{ij} نرخ همپوشانی بین دو مجموعه است و L_i نیز p / اول از مجموعه مرتب شده i ام و L_j برابر است با p / اول از مجموعه مرتب شده j ام. صورت کسر برابر است با تعداد اعضای اشتراک این دو مجموعه. مخرج کسر نیز برابر است با تعداد آزمایش‌های متفاوتی که در p / اول از کل k مجموعه مرتب شده وجود دارند.

در ادامه لازم است به‌ازای هر سنج معلوم شود که این سنج با سایر سنج‌ها به صورت میانگین چقدر همپوشانی یا هم‌بستگی دارد.

را با اولویت‌دهی تصادفی نیز مقایسه می‌کنند: روشی است که آزمایش‌ها را به صورت تصادفی در یک ترتیب قرار می‌دهد.

اکنون APFD مجموعه آزمون‌های مرتب‌شده توسط سنج‌ها و مجموعه آزمون‌های مرتب شده توسط فنون کنترلی بر روی نسخه‌های خطا دار مذکور، محاسبه می‌شوند. سپس مقادیر APFD حاصل از مجموعه آزمون‌های مرتب‌شده توسط سنج‌ها با استفاده از آزمون فرض آماری با مقادیر APFD فن کنترلی مقایسه می‌شوند. این کار با استفاده از الگوریتمی که در شکل (۲) آمده است انجام می‌شود.

Algorithm: Statistical Testing

```

Input: list of APFD for metric technique and control technique
Output: recognize which one is better
procedure STATISTICALTESTING (metricAPFD, controlAPFD)
1: if Kolmogrov-Smirnov-test(metricAPFD, controlAPFD) is normal
2: if p-value of t-test (metricAPFD, controlAPFD) < 0.05
3: reject the null hypothesis
4: return p-value and the better one
5: else
6: return the means are equal
7: else
8: if p-value of Wilcoxon-test (metricAPFD, controlAPFD) < 0.05
9: reject the null hypothesis
10: return p-value and the better one
11: else
12: return the means are equal
13: end if
14: end if
15: end procedure

```

شکل ۲: الگوریتم آزمون فرض آماری

برای انجام آزمون فرض آماری، ابتدا لازم است تا سطح معنی‌داری آماری تعیین شود. مطابق با عرف ادبیات، سطح معنی‌داری آماری ۰/۰۵ در نظر گرفته شده است. سپس مطابق شکل ۲، الگوریتم آزمون فرض، دو لیست از مقادیر APFD فن سنج و فن کنترلی را دریافت می‌کند. ابتدا با استفاده از آزمون کولموگروف-اسمیرنوف [۳۷] مشخص می‌شود که آیا هر دو از توزیع نرمال برخوردار هستند یا خیر. در واقع فرض صفر برای این آزمون، توزیع نرمال داشتن مقادیر دو لیست است. در صورت تأیید فرض صفر، با استفاده از آزمون t میانگین دو لیست مقایسه می‌شوند. در این آزمون، فرض صفر برابر بودن میانگین‌های دو لیست است. در صورت کوچک‌تر بودن p -value از سطح معنی‌داری ۰/۰۵، فرض صفر رد می‌شود و مشخص می‌شود که کدام یک از دو لیست میانگین بهتری دارند. در غیر این صورت، تفاوت معناداری بین میانگین‌ها وجود ندارد و فرض صفر پذیرفته می‌شود.

اگر فرض صفر برای آزمون کولموگروف-اسمیرنوف رد شود، معنایش این است که داده‌ها توزیع نرمال ندارند و باید از روش‌های آزمون ناپارامتری مثل آزمون ویلکاکسون [۳۷] برای مقایسه میانگین‌ها استفاده شود. در اینجا نیز فرض صفر برابر بودن میانگین‌های دو لیست است. در صورت کوچک‌تر بودن p -value از سطح معنی‌داری ۰/۰۵، فرض صفر رد می‌شود و مشخص می‌شود که کدام یک از دو لیست میانگین بهتری دارند. در غیر این صورت، تفاوت معناداری بین میانگین‌ها وجود ندارد و فرض صفر پذیرفته می‌شود. بنابراین مشخص می‌شود که کدام یک از سنج‌ها توانسته‌اند مجموعه آزمون مرتب شده بهتری را نسبت به فن کنترلی تولید کنند.

سؤال چهارم: استفاده از سنج‌های پیشنهادی در اولویت‌دهی آزمایش‌ها در چه شرایطی خوب جواب می‌دهد و در چه شرایطی گمراه‌کننده است؟

گام‌های ارزیابی فن پیشنهادی عبارت‌اند از:

جمع‌آوری اطلاعات پوشش آزمون: این کار با استفاده از ابزار جمع‌آوری پوشش کد انجام می‌شود. برای جمع‌آوری اطلاعات پوشش یک برنامه باید تغییراتی در فایل تنظیمات برنامه اعمال شود تا ابزار بتواند اطلاعات پوشش را از برنامه جمع‌آوری کند. ابزار، آزمایش‌ها را بر روی برنامه اجرا می‌کند و سپس اطلاعات پوشش در سطح کلاس و متد به‌زای هر برنامه ذخیره می‌گردد. ابزار و تنظیمات موردنیاز برای انجام این کار در ادامه ذکر می‌شود.

جمع‌آوری اطلاعات سنج‌های کد: این کار با استفاده از ابزار تحلیل ایستا انجام می‌شود. بدین منظور از طریق برنامه‌سازی به آن ابزار درخواست داده می‌شود که اطلاعات سنج‌های برنامه را در اختیارمان قرار دهد. سپس این اطلاعات به‌زای هر برنامه در سطح متد و کلاس ذخیره می‌گردد. ابزار موردنیاز برای انجام این کار در بخش بعدی ذکر می‌شود.

ترکیب اطلاعات پوشش و اطلاعات سنج‌های: به همان روشی که در بخش روش پیشنهادی بیان شد، اطلاعات به‌دست‌آمده در دو مرحله قبل با یکدیگر ترکیب می‌شوند و نتیجه ذخیره می‌گردد.

ساخت ماتریس خطاها: این کار با استفاده از ابزار جهش انجام می‌گیرد. ابزار جهش با دریافت برنامه و مجموعه آزمون متناظر با آن، اقدام به کاشت خطا در برنامه می‌کند. سپس با اجرا کردن آزمایش‌ها بر روی کد خطادار، مشخص می‌کند که هر آزمایش کدام خطاهای کاشته شده را توانسته تشخیص دهد. این اطلاعات را در قالب ماتریسی که در بخش روش پیشنهادی بیان شد، ذخیره می‌شوند.

اولویت‌دهی آزمایش‌ها براساس سنج‌ها: اکنون براساس اطلاعاتی که در گام قبل به‌دست آمد و طبق روشی که در روش پیشنهادی بیان شد، به ازای هر کدام از سنج‌ها یک مجموعه آزمون مرتب‌شده وجود دارد.

مقایسه عملکرد مجموعه آزمون‌های مرتب‌شده براساس سنج‌ها با فن کنترلی (مبنای مقایسه): در این گام به‌زای هر برنامه، ۵۰۰ خطا با استفاده از ابزار جهش کاشته می‌شود. سپس نسخه‌های خطادار از آخرین نسخه هر برنامه از طریق گروه‌بندی کردن هر پنج خطا با یکدیگر، ایجاد می‌شوند؛ یعنی برای هر برنامه به‌طور تصادفی این ۵۰۰ خطا را در ۱۰۰ گروه خطا که هر کدام شامل پنج خطای تصادفی است، تقسیم‌بندی می‌شوند. سپس مجموعه آزمون‌های مرتب‌شده از فنون مختلف بر روی این ۱۰۰ نسخه خطادار از هر برنامه اعمال می‌شوند و معیار APFD محاسبه می‌شود. اکنون با انجام آزمون فرض آماری و با استفاده از الگوریتم آزمون فرض، به مقایسه نتایج این دو فن پرداخته می‌شود.

بدین‌منظور میزان هم‌بستگی هر سنج نسبت به سایر سنج‌ها طبق رابطه (۴) محاسبه می‌شود.

$$o_rate_{avg}(j) = \frac{1}{k-1} \sum_{i=1,2,\dots,k, i \neq j} o_rate_{ij} \quad (4)$$

حال سنج‌ای که مقدار o_rate_{avg} آن به عدد یک نزدیک‌تر باشد، در واقع اشتراکی از سایر سنج‌ها است و رأی مستقلی در تصمیم‌گیری نهایی ندارد. وزن هر سنج طبق رابطه (۵) به‌دست می‌آید.

$$w_j = 1 - o_rate_{avg}(j) \quad (5)$$

طبق رابطه بالا آن سنج‌ای که هم‌بستگی کمتری با بقیه دارد، در اولویت‌دهی نهایی تعیین‌کننده‌تر است و وزن بیشتری می‌گیرد. اکنون وزن هر کدام از سنج‌ها به‌دست آمده است. حال طبق رابطه، امتیازی که روش ترکیب خطی به آزمایش‌ها می‌دهد قابل محاسبه است. اما قبل از آن لازم است تا امتیازی که توسط سنج‌ها به آزمایش‌ها داده شده‌اند، نرمال‌سازی شوند. برای نرمال‌سازی از روش خطی صفر-یک که در رابطه (۶) آمده است، استفاده می‌شود [۵].

$$s_{ij} = \frac{MetricScore(i, j) - \min(MetricScore(j))}{\max(MetricScore(j)) - \min(MetricScore(j))} \quad (6)$$

که $MetricScore(i, j)$ امتیازی است که سنج j به آزمایش i داده و $\min(MetricScore(j))$ برابر است با حداقل امتیازی که سنج j به آزمایش‌ها داده است. همچنین $\max(MetricScore(j))$ برابر است با حداکثر امتیازی که سنج j به آزمایش‌ها داده است و S_{ij} نیز برابر است با امتیاز نرمال‌شده‌ای که سنج j به آزمایش i داده است.

حال با استفاده از رابطه (۲)، امتیاز آزمایش‌ها توسط روش ترکیب خطی به‌دست می‌آید. اکنون آزمایش‌ها با استفاده از امتیازهای به‌دست‌آمده اولویت‌دهی می‌شوند و مجموعه آزمون مرتب‌شده با k مجموعه دیگری که مربوط به k سنج برتر بودند مقایسه می‌شوند.

۴- ارزیابی

به‌منظور ارزیابی فن پیشنهادی، ابتدا الگوریتم‌های معرفی‌شده پیاده‌سازی می‌شوند و آزمایش‌هایی انجام می‌شوند. در آزمایش‌ها به دنبال پاسخ به سؤال‌های تحقیقی زیر هستیم.

سؤال اول: اولویت‌دهی با سنج‌ها در مقایسه با روش‌های کنترلی (مبنای مقایسه) چگونه عمل می‌کند؟

سؤال دوم: سنج‌های نرم‌افزاری براساس ریزدانگی‌شان چه تأثیری در اولویت‌دهی دارند؟

سؤال سوم: چگونه می‌توان اطلاعات این سنج‌ها را به‌منظور بهبود اثربخشی، ترکیب کرد و این ترکیب چگونه عمل می‌کند؟

شده‌است. برای یکپارچه کردن اطلاعات به‌دست‌آمده از ابزارها و پیاده‌سازی روش پیشنهادی از جاوا در محیط IntelliJ IDEA نسخه ۱۲.۰۱۶ استفاده شده است. پیاده‌سازی و اجرای برنامه‌های این ارزیابی همگی توسط ماشین با مشخصات زیر انجام شده است: سیستم عامل ویندوز ۱۰، پردازنده چهار هسته Intel Core i7-4710HQ 2.50GHz و حافظه ۱۲ گیگابایت.

۲-۴- برنامه‌های محک و ویژگی‌های آن‌ها

ارزیابی‌های ما بر روی هفت برنامه جاوا که به‌طور گسترده در تحقیقات آزمون بازگشت مورد استفاده قرار گرفته‌اند [۳، ۴، ۱۲-۱۰]، انجام شده‌اند. برای هر برنامه، مجموعه آزمون JUnit متناظر، اوراکل‌ها، راه‌اندازها و اسکریپت‌های لازم توسط محققان فراهم شده و در مخزن گیت‌هاب^{۱۷} موجود هستند. تعداد آزمایش‌ها و تعداد خطوط مربوط به هر کدام از برنامه‌ها در جدول (۳) آمده است.

جدول ۳: ویژگی‌های برنامه‌های محک

برنامه	apns	assertj	jasmine	jopt	la4j	metrics	scribe
آزمایش‌ها	۸۵	۲۴۶۲	۱۰۲	۷۲۷	۶۲۵	۱۴۹	۹۸
خطوط	۱۳۶۲	۵۵۴۴۲	۱۶۴۰	۶۶۳۶	۸۰۹۴	۱۱۴۷۷	۲۴۹۷

۳-۴- نتایج آزمایش‌ها

نتایج حاصل از اجرای الگوریتم آزمون فرض روی مقادیر APFD برنامه‌های محک در جدول (۴) آمده است. هر سلول این جدول حاصل مقایسه اولویت‌دهی با سنج در سطح متناظر و فن پوشش کامل را نشان می‌دهد. در این جدول سطرها ۱ تا ۱۱ مربوط به سنج‌های سطح متد، از شماره ۱۲ تا ۲۷ مربوط به سنج‌های سطح کلاس و سطرها ۲۸ مربوط به فن اولویت‌دهی تصادفی است. هر سلول از این دو جدول حاوی چهار نوع اطلاعات است. در صورتی که فن مورد نظر از فن کنترلی بهتر باشد، با علامت ✓، در صورتی که بهتر نباشد با علامت ✗ و در صورتی که تفاوت قابل توجهی بین این دو فن وجود نداشته باشد با علامت O مشخص شده است. در صورتی که آزمون فرض آماری با ویلکاکسون با شد، علامت W و در صورتی که آزمون t با شد با حرف T مشخص شده است. مقدار *p-value* نیز در هر سلول آمده است. همچنین اگر مقدار *p-value* از سطح معنی‌داری ۰/۰۵ کمتر باشد، سلول به رنگ تیره در آمده است.

۴-۴- ترکیب سنج‌های برتر

در این قسمت، سنج‌های برتری که در مرحله قبل شناسایی شدند با روش ترکیب خطی، ترکیب می‌شوند. نتایج حاصل به‌ازای ۷ برنامه در قسمت‌های ۱ تا ۷ شکل (۳) در قالب نمودار جعبه‌ای آمده‌اند. به‌ازای هر برنامه، اسامی سنج‌های برتر به‌ازای آن برنامه در زیر شکل از چپ به راست آمده‌اند. سنج‌های سطح کلاس با (C) مشخص شده‌اند. آخرین جعبه در هر نمودار که در سمت راست قرار دارد و با نام LC

ترکیب سنج‌های برتر از طریق روش ترکیب خطی: در این مرحله ابتدا لازم است تا سنج‌های برتر به‌ازای هر برنامه محاسبه شوند؛ سنج‌هایی که مجموعه آزمون‌هایی را تولید کرده‌اند که از لحاظ APFD برتر بوده‌اند. تعداد این سنج‌ها به‌ازای هر برنامه به‌صورت پارامتریک تعیین می‌شود. سپس از طریق روش ترکیب خطی، این سنج‌ها به‌منظور افزایش مقدار APFD ترکیب می‌شوند. مقدار پارامتر *p* برای رابطه (۳)، برای تمامی برنامه‌ها برابر ۵۰٪ در نظر گرفته شده است. در پایان مجموعه آزمون مرتب‌شده حاصل از ترکیب سنج‌ها با سنج‌های برتر مقایسه می‌شود.

۱-۴- ابزارهای مورد استفاده

در این تحقیق برای جمع‌آوری اطلاعات پوشش، از ابزار پوشش‌کد Jacoco^{۱۲} استفاده شده است که کتابخانه استخراج پوشش‌کد برای جاوا می‌باشد. این ابزار از طریق فراکدگذاری بر روی کد میانی برنامه جاوا، اطلاعات پوشش برنامه را استخراج می‌نماید. همان‌طور که گفته شد برای جمع‌آوری اطلاعات پوشش یک برنامه باید تغییراتی در فایل تنظیمات برنامه اعمال شود تا ابزار بتواند اطلاعات پوشش را از برنامه جمع‌آوری کند. از آن‌جا که برنامه‌های محک مورد استفاده در این تحقیق از Maven^{۱۳} به‌عنوان ابزار ساخت و مدیریت پروژه استفاده می‌کنند، برای این که Jacoco بتواند راه‌اندازی شود، باید افزونه Jacoco در فایل pom.xml هر پروژه پیکربندی شود.

برای جمع‌آوری اطلاعات سنج‌های از ابزار Understand^{۱۴} استفاده شده است. این ابزار برای محاسبه سنج‌های کد مناسب است که در مقالات متعددی از جمله [۹-۶] برای محاسبه سنج‌ها در حوزه‌های مختلف استفاده شده است.

برای ایجاد خطا در برنامه‌ها و ساختن ماتریس خطاها، از ابزار PIT^{۱۵} استفاده شده است. این ابزار در تحقیقات زیادی به‌منظور انجام آزمون جهش به کار گرفته شده است. مزیت این ابزار این است که برای یافتن سریع‌تر آزمایش‌هایی که خطاهای کاشته‌شده را تشخیص می‌دهند، از ابزار پوشش‌کد Jacoco استفاده می‌کند؛ بدان معنا که ابتدا با استفاده از Jacoco مشخص می‌کند که هر آزمایش چه خطوی از برنامه را پوشش داده است. سپس به‌ازای هر خطای کاشته‌شده، تنها آن آزمایش‌هایی را اجرا می‌کند که خط متناظری که آن خطا در آن خط کاشته شده را پوشش می‌دهند. این کار باعث تسریع در عملکرد آزمون جهش می‌شود. در تحقیق ما نیز از خطاهای کاشته‌شده آماده در تحقیق لو و همکاران [۳] استفاده شده است.

برای جمع‌آوری داده‌های موردنیاز برای الگوریتم آزمون فرض از نرم‌افزار IBM SPSS Statistics 23^{۱۶} استفاده شده است. نرم‌افزار IBM SPSS از جمله نرم‌افزارهایی است که برای تحلیل‌های آماری به‌صورت بسیار گسترده‌ای استفاده می‌شود. علت استفاده از این نرم‌افزار توانایی آن در انجام تحلیل‌های آماری و خروجی‌های مناسبی است که در اختیار کاربر قرار می‌دهد. همچنین برای پیاده‌سازی الگوریتم آزمون فرض از زبان برنامه‌سازی VBA و نرم‌افزار اکسل نسخه ۲۰۱۳ استفاده

هستند، موفق‌تر عمل می‌کنند. این نتیجه حاکی از این است که هر چه ریزدانه‌گی سنج‌ها بیشتر باشد، سنج‌ها بیش‌تر می‌توانند آزمایش‌ها را از یکدیگر متمایز کنند.

بدین ترتیب پاسخ سؤال دوم تحقیق این است که باتوجه به وجود این عملکرد در کلیه برنامه‌های مورد مطالعه، عملکرد سنج‌ها در سطح متد پایدارتر است و نتایج بهتری را به‌همراه دارد.

با توجه به نتایج شکل (۳) از ترکیب سنج‌های برتر، خروجی‌ها تحلیل می‌شوند. برای برنامه‌های `apns`، `la4j`، `jasmine`، `assertj` و `metrics` مقدار APFD برای ترکیب سنج‌ها به‌طور میانگین به‌ترتیب ۰/۳، ۰/۲، ۰/۲، ۰/۴ و ۰/۴ به‌نسبت برترین سنج، بهبود داشته است. برای دو برنامه `jopt` و `scribe`، ترکیب سنج‌ها باعث بهبود در مقدار میانگین APFD نمی‌شود.

بنابراین پاسخ سؤال سوم تحقیق این است که سنج‌های برتر با استفاده از روش ترکیب خطی با یکدیگر ترکیب می‌شوند.

طبق آزمایش‌های انجام‌شده، مقدار APFD مجموعه آزمون اولویت‌دهی شده حاصل در بیش‌تر موارد نسبت به مجموعه آزمون‌هایی که توسط سنج‌های برتر اولویت‌دهی شده‌اند، بهبود داشته است. عدم بهبود مقدار APFD در برخی موارد به‌علت عدم قطعیت فنون امتزاج داده در بهبود اثربخشی می‌باشد. عدم قطعیت یکی از ویژگی‌های عمومی روش‌های امتزاج داده است [۵]؛ یعنی در برخی موارد اثربخشی داده‌های امتزاج‌شده توسط یک روش امتزاج داده بهتر از بهترین آن‌هاست. در برخی موارد نیز اثربخشی داده‌های امتزاج‌شده توسط همان روش امتزاج داده، از بدترین داده‌ها نیز بهتر نیست.

۴-۶- شناخت عوامل زمینه‌ای در موفقیت روش

در این بخش، بررسی می‌شود که استفاده از سنج‌های کد در اولویت‌دهی آزمایش‌ها در چه شرایطی خوب جواب می‌دهد یا گمراه‌کننده می‌شود. بدین‌منظور، نحوه امتیازدهی سنج‌ها به آزمایش‌ها بررسی می‌شوند. برای مثال سنج `MaxInheritanceTree` (عمق وراثت) را در نظر بگیرید. این سنج در برنامه `apns` عملکرد قابل‌قبولی نداشته است. با بررسی امتیازهایی که این سنج به آزمایش‌ها داده است، معلوم شد که کلاس‌های این برنامه عمق وراثت چندانی ندارند و امتیازهایی که این سنج به آزمایش‌ها داده است صفر، یک یا دو است. بنابراین امتیازهای آزمایش‌ها تقریباً یکسان است و تعداد گره‌ها افزایش می‌یابد. در نتیجه اولویت‌دهی با استفاده از این سنج خیلی شبیه به اولویت‌دهی تصادفی آزمایش‌ها می‌باشد.

همین سنج در برنامه `la4j` مورد بررسی قرار می‌گیرد. این سنج توانسته عملکرد قابل‌قبولی را در اولویت‌دهی آزمایش‌ها برای این برنامه داشته باشد. با بررسی امتیازهایی که این سنج به آزمایش‌ها داده است معلوم شد که این برنامه عمق وراثت بیش‌تری دارد و کلاس‌ها مقادیر

مشخص شده‌است، مربوط به نتیجه ترکیب سنج‌های برتر می‌باشد. هر جعبه در هر نمودار پراکندگی ۱۰۰ مقدار APFD حاصل از اولویت‌دهی براساس یک سنج را در چارک‌ها نشان می‌دهد. چارک دوم و سوم در مربع/مستطیل وسط هر جعبه نشان داده می‌شود و خط درون آن نشان‌دهنده میانه داده‌هاست یا همان انتهای چارک دوم. دم پایین مربوط است به چارک اول داده‌ها و دم بالا مربوط است به چارک چهارم داده‌ها. کوچک‌تر بودن عرض مربع/مستطیل میانی و در کل کوچک‌تر بودن فاصله دو دم دلالت بر مرکزیت بیش‌تر داده‌ها دارد.

همچنین، قسمت ۸ در شکل (۳)، نمودار Q-Q را به‌ازای دو سنج `LC` و `MaxCyclomatic(C)` در برنامه `scribe` نشان می‌دهد. محور عمودی در این نمودار به مقادیر APFD های حاصل از ترکیب سنج‌های برتر و محور افقی به مقادیر APFD های سنج `MaxCyclomatic(C)` اختصاص دارد. آن‌چنان‌که دیده می‌شود، در محدوده ۰/۷ تا ۰/۹ عملکرد ترکیب سنج‌ها و سنج منفرد تقریباً یکسان بوده است. اما در محدوده ۰/۵ تا ۰/۷ گاهی عملکرد یکی از دو حالت بهتر یا بدتر می‌گردد. برای نمونه، در قسمت ۷ از شکل (۳) و در مورد دو جعبه سمت راست، مشاهده می‌شود که پراکندگی مقادیر APFD های سمت راست‌ترین جعبه اندکی کمتر است و میانه آن نیز اندکی بالاتر از جعبه سمت چپ آن است.

۴-۵- بحث در نتایج

جدول (۴) نشان می‌دهد که برای بیشتر برنامه‌ها، سنج‌ها به نسبت فن پوشش کامل، بهتر عمل می‌کنند. ولی نحوه عملکرد سنج‌ها نسبت به فن تصادفی در مقایسه با فن پوشش کامل بهتر می‌باشد. برای پی بردن به‌علت عدم عملکرد بهتر سنج‌ها نسبت به فن کنترلی، امتیازهایی که سنج‌ها به آزمایش‌ها داده‌اند، بررسی گردید.

در مواردی که سنج‌ها نسبت به فن کنترلی عملکرد بهتری نداشتند، مشاهده شد که تعداد حالت‌های تساوی خیلی زیاد است؛ به‌عبارت بهتر، تعداد زیادی آزمایش هستند که سنج‌ها، امتیاز یکسانی به آن‌ها داده‌اند. درواقع چنین سنج‌هایی با ساختار و ویژگی‌های این برنامه تناسب ندارند. برای مثال سنج عمق وراثت برای برنامه‌ای که ویژگی وراثتی زیادی ندارد، مناسب نیست. چنین سنج‌هایی به‌خوبی قادر نیستند تا این آزمایش‌ها را از هم متمایز کنند. در نتیجه چنین آزمایش‌هایی به‌صورت تصادفی مرتب می‌شوند و باعث کاهش اثربخشی مجموعه آزمون می‌گردند. این مسئله در کار اقبالی و تحویل‌داری [۱۳] نیز وجود داشت که محققان به این نتیجه رسیده بودند که در صورت افزایش تعداد حالت تساوی، اثربخشی فنون اولویت‌دهی کاهش می‌یابد.

بنابراین پاسخ سؤال اول تحقیق این است که اولویت‌دهی با سنج‌ها عملکرد بهتر و پایدارتری نسبت به فن کنترلی (مبنای مقایسه) دارد.

در کلیه برنامه‌ها، سنج‌های سطح متد در مجموع عملکرد بهتری نسبت به سنج‌های سطح کلاس داشته‌اند. این مشاهده نشان می‌دهد که سنج‌های سطح متد در تشخیص مکان‌هایی از برنامه که خطاخیزتر

همچنین در مقیاس صنعتی برای شناسایی سنج‌هایی که بتوانند تمایز خوبی بین آزمایش‌ها قائل شوند، می‌توان از نظرات توسعه‌دهندگان و خبرگان سیستم کمک گرفت. در واقع می‌توان از نظرات آنان جویا شد که پیچیدگی سیستم از چه جنبه‌ای بیشتر است. در نتیجه می‌توان فرآیند اولویت‌دهی را به سمتی سوق داد که مجموعه آزمون مرتب‌شده‌ای تولید شود که هرچه سریع‌تر خطاها را تشخیص دهد.

۴-۷- تهدیدهای اعتبار نتایج

در انجام آزمایش‌های این تحقیق از خطاهای کاشته شده‌ای که توسط ابزار PIT تولید شده‌اند، استفاده شده‌است. یک تهدید احتمالی بر آزمایش‌ها این است که خطاهای کاشته شده بیان‌گر خصوصیات طبیعی خطاهای واقعی نباشند. برای کاهش این تهدید، طبق راهبردهایی که تحقیقات در این حوزه پیشنهاد کرده بودند، از ۵۰۰ خطا به‌طور تصادفی، ۱۰۰ نسخه خطا دار ساخته شد.

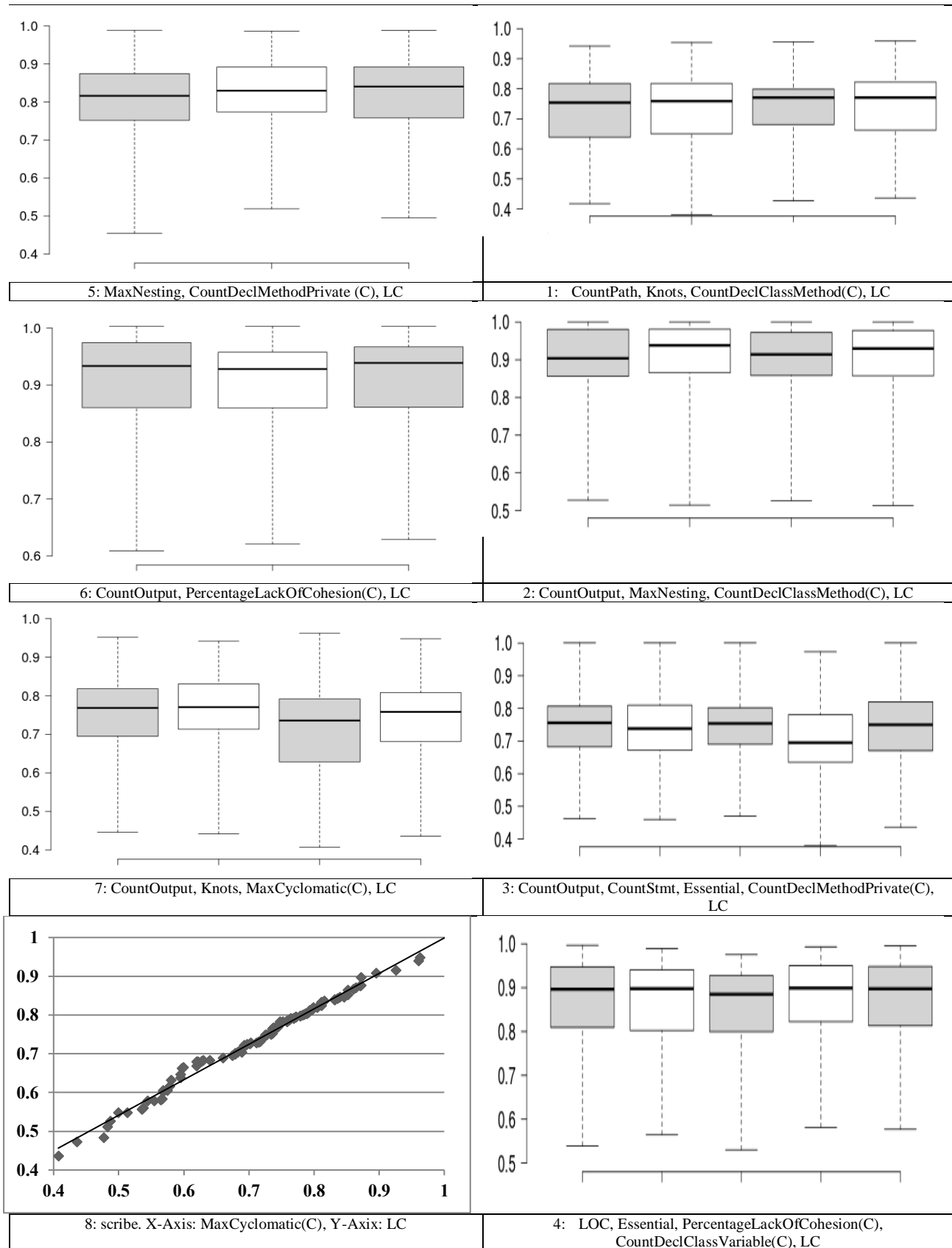
بیشتر و متفاوت‌تری از این سنج‌ها دارند. بنابراین این سنج‌ها توانسته به‌خوبی آزمایش‌ها را از یکدیگر تفکیک کند و تعداد حالت‌های تساوی را کاهش داده است. به همین دلیل اثربخشی این سنج‌ها در اولویت‌دهی آزمایش‌ها قابل قبول بوده است.

در واقع هر سنج از یک جنبه‌ای به نرم‌افزار نگاه می‌کند و از آن جنبه به آزمایش‌ها امتیاز می‌دهد. حال اگر آن جنبه به آزمایش‌ها امتیاز یکسانی بدهد و نتواند بین آزمایش‌ها تفاوت قائل بشود، باعث ایجاد حالت تساوی گمراه‌کنندگی در فرآیند اولویت‌دهی می‌شود.

بنابراین پاسخ سؤال چهارم این است که در شرایطی سنج‌ها خوب جواب می‌دهند که بتوانند به‌خوبی آزمایش‌ها را از یکدیگر متمایز کنند. در صورتی که امتیاز یکسانی به آزمایش‌ها بدهند، عملکرد اولویت‌دهی احتمالاً شبیه فن تصادفی می‌شود. در نتیجه اطلاعات چنین سنج‌هایی برای اولویت‌دهی گمراه‌کننده است.

جدول ۴: نتایج حاصل از انجام آزمون فرض آماری برای مقایسه فن پیشنهادی با فن پوشش کامل

scribe	metrics	La4j	jopt	Jasmine	assertj	apns	سنج (فن) / برنامه محک	
X T0.002	✓ W0.012	✓ W0	X W0	✓ T0	OW0.428	✓ T0	CountInput	۱
✓ T0	X W0	✓ T0	✓ W0.001	✓ T0	✓ W0	✓ W0	CountOutput	۲
OW0.208	✓ W0.004	✓ T0	X W0	✓ T0.001	✓ W0.001	✓ W0	CountPath	۳
OW0.45	OW0.62	✓ T0	X W0.011	X T0	OW0.07	✓ T0.025	CountPathLog	۴
✓ W0.005	X W0.029	✓ T0	✓ W0.012	✓ T0	✓ W0	✓ W0	CountStmt (LOC)	۵
✓ T0.032	OW0.418	✓ T0	OW0.119	✓ T0	✓ W0.006	✓ W0	Essential	۶
✓ W0.012	X W0	✓ T0	X W0	X T0	✓ W0.001	✓ W0	Knots	۷
OT0.595	X W0.004	✓ T0	X W0	X T0	✓ W0.005	X W0.044	MinEssentialKnots	۸
OW0.24	X W0.001	✓ T0	X W0.033	X T0	✓ W0	✓ T0.013	MaxNesting	۹
OW0.198	OW0.49	✓ T0	OW0.077	✓ T0.001	✓ W0.002	✓ W0	Cyclomatic	۱۰
OW0.08	OW0.476	✓ T0	X W0	✓ T0.002	✓ W0.002	✓ T0	CyclomaticStrict	۱۱
X T0	OW0.247	✓ T0	X W0.033	X T0	OW0.665	X T0	PercentLackOfCohesion	۱۲
X T0	OW0.576	✓ T0	X W0	X T0	X T0	X T0	MaxInheritanceTree	۱۳
X T0.025	OW0.345	✓ T0	X W0.001	X T0	OW0.282	X T0	CountClassDerived	۱۴
X T0.023	OW0.283	✓ T0	X W0	X T0	OW0.536	OT0.675	CountClassCoupled	۱۵
X T0.012	OW0.969	✓ T0	X W0	X T0	OT0.286	OT0.074	CountClassBase	۱۶
X T0.002	X W0.002	✓ T0	X W0	X T0	✓ W0	✓ W0	CountDeclClassMethod	۱۷
X W0	OW0.976	✓ T0	OW0.391	X T0	OW0.643	X T0	CountDeclClassVariable	۱۸
OT0.068	OW0.827	✓ T0	X W0	X T0.003	✓ W0.037	OT0.115	AvgCyclomaticStrict	۱۹
OW0.393	OW0.253	✓ T0	X W0	X T0	✓ W0.049	OT0.57	AvgLineCode	۲۰
X W0	OW0.244	✓ T0	X W0	X T0	OT0.128	X T0	CountDeclMethodAll	۲۱
X T0	OW0.211	✓ W0	X W0	X T0	X T0	X T0	CountDeclMethodDefault	۲۲
X W0.004	OW0.236	✓ T0	X W0	OT0.274	OW0.606	OW0.391	CountDeclMethodPrivate	۲۳
OW0.204	OW0.819	✓ T0	X W0	X T0	✓ W0.005	OT0.157	MaxCyclomatic	۲۴
OT0.126	OW0.558	✓ W0	X W0	X T0	✓ W0.005	OT0.735	MaxEssential	۲۵
X W0.001	X W0.041	✓ T0	X W0	X T0	OW0.124	✓ T0.024	SumCyclomatic	۲۶
X W0	X W0.01	✓ T0	X W0	X T0	OW0.213	OT0.211	SumEssential	۲۷
X T0	X W0	✓ T0	OW0.093	X T0	X W0	OT0.705	تصادفی	



شکل ۳: نتایج APFD به‌ازای سنجه‌های برتر و ترکیب آن‌ها. شماره‌های ۱ تا ۷ به ترتیب برنامه‌های `apns`, `assertj`, `jasmine`, `jopt`, `ja4j`، `metrics` و `scribe` هستند. نمودار مربوط به هر برنامه جعبه‌هایی از چپ به راست دارد. هر جعبه مربوط می‌شود به اولویت‌دهی با کمک اطلاعات یک سنجه که زیر هر شکل به ترتیب از چپ به راست مشخص شده‌اند. شماره ۸ نیز نمودار Q-Q از مقایسه دو سنجه را برای برنامه `scribe` نشان می‌دهد.

۵- کارهای مرتبط

می‌شوند که آزمایش‌های همسایه آن‌ها تا جای ممکن تنوع بیشتری داشته‌باشند.

علیقاردا شی و زارع [۳۸] با استفاده از انتخاب پیش‌روی ویژگی از یادگیری ماشینی، روشی برای پیش‌بینی خطای نرم‌افزار ارائه کردند. مشابه روش این مقاله، هدف مقاله مذکور نیز شنا سایی خطاهاست با این تفاوت که محققان سعی می‌کنند ماژول‌های مستعد خطا را شناسایی نمایند. در رابطه با کمینه‌سازی مجموعه آزمون پس‌نمایی، رافع و اسفندیاری [۳۹] روشی با ترکیب الگوریتم جستجوی تپه‌نوردی و جستجوی خفاش ارائه کردند. کمینه‌سازی روشی است که در کنار اولویت‌دهی آزمایش‌ها (که هدف این مقاله بوده است) برای بهبود اثربخشی آزمون نرم‌افزار مورد استفاده قرار می‌گیرد.

۶- نتیجه‌گیری

یکی از چالش‌های فنون اولویت‌دهی آزمایش‌ها حین آزمون بازگشت، تخمین درست قدرت آشکارسازی خطای آزمایش‌هاست. با توجه به اثربخشی سنج‌های کد در پیش‌بینی خطاها، پیشنهاد شد که از این اطلاعات برای تخمین میزان اثربخشی آزمایش‌ها در کشف خطا استفاده شود. بر این اساس فن اولویت‌دهی جدیدی بر اساس امتزاج داده روی اطلاعات سنج‌های کد ارائه شد و برای اعتبار سنجی آن، آزمایش‌هایی ترتیب داده شد. نتایج آزمایش‌ها نشان داد: (۱) سنج‌های نرم‌افزار در اولویت‌دهی آزمایش‌ها اثربخش‌اند؛ (۲) برای گرفتن نتایج بهتر، باید سنج‌های برتر با هم ترکیب شوند؛ (۳) استفاده از سنج‌ها وقتی اثربخش‌تر می‌شوند که امتیازهایشان بتوانند تک‌تک آزمایش‌ها را از هم متمایز نمایند. در آینده تصمیم این است که فن پیشنهادی روی برنامه‌های بزرگ‌تر و زبان‌های دیگر رایج نیز مورد ارزیابی قرار گیرد.

مراجع

- [1] Z. Li, M. Harman, and R. M. Hierons, "Search algorithms for regression test case prioritization," *IEEE Trans. Softw. Eng.*, vol. 33, no. 4, 2007.
- [2] L. Zhang, D. Hao, L. Zhang, G. Rothermel, and H. Mei, "Bridging the gap between the total and additional test-case prioritization strategies," *Proc. - Int. Conf. Softw. Eng.*, pp. 192-201, 2013.
- [3] Y. Lu, Y. Lou, S. Cheng, L. Zhang, D. Hao, Y. Zhou, and L. Zhang, "How does regression test prioritization perform in real-world software evolution?," *Proc. 38th Int. Conf. Softw. Eng. - ICSE '16*, pp. 535-546, 2016.
- [4] Q. Luo, K. Moran, and D. Poshyanyk, "A large-scale empirical comparison of static and dynamic test case prioritization techniques," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 559-570, 2016.
- [5] S. Wu, *Data Fusion in Information Retrieval*. Springer Berlin Heidelberg, 2012.
- [6] T. Lee, J. Nam, D. Han, S. Kim, and H. Peter In, "Developer Micro Interaction Metrics for Software Defect Prediction," *IEEE Trans. Softw. Eng.*, vol. 42, no. 11, pp. 1015-1035, 2016.
- [7] F. Rahman and P. Devanbu, "How, and why, process metrics are better," in *Proceedings of the 2013*

عرفین و همکاران [۲۸]، از اطلاعات نیازمندی‌ها برای بهبود اولویت‌دهی آزمایش‌ها استفاده کرده‌اند. ریکانت^{۱۸} و همکاران [۲۹] گزارش کرده‌اند که آزمایش‌های اولویت‌دهی شده بر اساس اهمیت و خطاخیزی نیازمندی‌ها، قادر بوده خطاهای دشوار را زودتر کشف نماید.

توماس و همکاران [۳۰]، اولویت‌دهی آزمایش‌ها را به صورت ایستا و با استفاده از مدل‌های موضوعی انجام داده‌اند. با وجودی که این روش سریع و ساده است، اما خطر نادیده گرفتن جنبه‌های مهم آزمایش‌ها و خودکارسازی کم‌روشی از مشکلات آن هستند.

بنابر تحقیقات رودرمل و همکاران [۳۱]، اکثر فنون اولویت‌دهی موجود، فرضشان این بوده است که آزمایش‌های که پوشش کد بیشتری دارد، احتمالاً خطاهای بیشتری را در برنامه تحت آزمون پیدا می‌کند. متأسفانه در بعضی موارد، پوشش کد پیش‌بینی‌کننده خوبی از قابلیت تشخیص خطای آزمایش نیست. همچنین تحقیقات کمی بر روی کدهایی که کمتر مورد آزمون قرار می‌گیرند، انجام شده است. چنین کدهایی اغلب خطاخیزی بالایی دارند و آزمون‌نشان برای صحت برنامه حیاتی است. بنابر اظهار اقبالی و همکاران [۱۳]، روش‌های رایج برای اولویت‌دهی، از اطلاعات آزمایش‌های قبلاً اجرا شده، مانند اطلاعات پوشش استفاده می‌کنند. اثربخشی این روش‌ها در صورت افزایش تعداد گره‌ها در گام‌های اولویت‌دهی، کاهش می‌یابد.

سایا و همکاران [۳۲]، روشی مبتنی بر بازیابی اطلاعات برای اولویت‌دهی ارائه کرده‌اند. پاندا و همکاران [۳۳] پیشنهاد کردند که تنها قسمت‌های تغییر کرده از برنامه را دوباره آزموده شود. بر این مبنا آن‌ها روشی ایستا برای اولویت‌دهی آزمایش‌ها در برنامه‌های شی‌گرا پیشنهاد کردند. بنابر گزارش محققان ماژولی که مقدار اتصال بالایی دارد، امکان وجود خطای بیشتری نسبت به ماژول‌های دیگر دارد. در نتیجه آزمایش‌های که ماژولی با مقدار اتصال بیشتری را اجرا می‌کند، خطاهای بیشتری را نسبت به دیگر آزمایش‌ها آشکار می‌نماید.

چودھاری^{۱۹} و همکاران [۳۵] مطالعه‌ای انجام دادند تا اثر سنج‌های مختلف را در بهبود کارایی مدل‌های پیش‌گویی خطا مورد بررسی قرار دهند. محققان با استفاده از اطلاعات سنج‌های تغییر و سنج‌های کد منبع، دسته‌بندی‌هایی به‌عنوان مدل پیش‌گویی خطا ساختند. نتایج آزمایش‌ها نشان داد که سنج‌های تغییر اثر مثبتی در عملکرد مدل پیش‌گویی خطا دارد.

چن^{۲۰} و همکاران [۳۶] روشی برای بهبود اثربخشی آزمون بازگشت در نرم‌افزارهای شی‌گرای ارائه کردند. این روش از فنون خوشه‌بندی روی اطلاعات جعبه سیاه استفاده می‌کند. محققان از تعداد اشیاء و متدها و رشته فراخوانی اشیاء و متدها به‌عنوان اطلاعات مورد استفاده در خوشه‌بندی استفاده کرده‌اند. در نتیجه آزمایش‌ها طوری اولویت‌دهی

- fault prediction," *IEEE Trans. Softw. Eng.*, vol. 31, no. 10, pp. 897–910, 2005.
- [25] R. P. L. Buse and W. Weimer, "The road not taken: Estimating path execution frequency statically," in *Proceedings of the 31st International Conference on Software Engineering*, 2009, pp. 144–154.
- [26] J. C. Sanchez, L. Williams, and E. M. Maximilien, "On the sustained use of a test-driven development practice at ibm," in *Agile Conference (AGILE), 2007*, 2007, pp. 5–14.
- [27] R. P. L. Buse and W. R. Weimer, "x," in *Proceedings of the 2008 international symposium on Software testing and analysis*, 2008, pp. 121–130.
- [28] M. J. Arafeen and H. Do, "Test Case Prioritization Using Requirements-Based Clustering," in *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation*, 2013, pp. 312–321.
- [29] H. Srikanth, L. Williams, and J. Osborne, "System test case prioritization of new and regression test cases," in *2005 International Symposium on Empirical Software Engineering, 2005.*, 2005, p. 10–pp.
- [30] S. W. Thomas, H. Hemmati, A. E. Hassan, and D. Blostein, "Static test case prioritization using topic models," *Empir. Softw. Eng.*, vol. 19, no. 1, pp. 182–212, 2014.
- [31] J. H. Kwon, I. Y. Ko, G. Rothermel, and M. Staats, "Test case prioritization based on information retrieval concepts," *Proc. - Asia-Pacific Softw. Eng. Conf. APSEC*, vol. 1, pp. 19–26, 2014.
- [32] R. K. Saha, L. Zhang, S. Khurshid, and D. E. Perry, "REPiR: An Information Retrieval based Approach for Regression Test Prioritization," in *37th International Conference on Software Engineering, Florence Italy, Mary*, 2015.
- [33] S. Panda, D. Munjal, and D. P. Mohapatra, "A Slice-Based Change Impact Analysis for Regression Test Case Prioritization of Object-Oriented Programs," vol. 2016, 2016.
- [34] Ammann, Paul, and Jeff Offutt. *Introduction to software testing*. Cambridge University Press, 2016.
- [35] Choudhary, Garvit Rajesh, Sandeep Kumar, Kuldeep Kumar, Alok Mishra, and Cagatay Catal. "Empirical analysis of change metrics for software fault prediction." *Computers & Electrical Engineering* 67, pp. 15-24, 2018.
- [36] Chen, Jinfu, Lili Zhu, Tsong Yueh Chen, Dave Towey, Fei-Ching Kuo, Rubing Huang, and Yuchi Guo. "Test case prioritization for object-oriented software: An adaptive random sequence approach based on clustering." *Journal of Systems and Software* 135, pp. 107-125, 2018.
- [37] Wohlin, Claes, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.
- [۳۸] فاطمه علیقارداشی، محمدعلی زارع چاهوکی، « تأثیر ترکیب روش‌های انتخاب ویژگی فیلتر و بسته‌بندی در بهبود پیش‌بینی اشکال نرم‌افزار»، *مجله مهندسی برق*، دوره ۴۷، شماره ۱، ۱۹۵–۱۸۳، دانشگاه تبریز، بهار ۱۳۹۶.
- [۳۹] وحید رافع، سجاد اسفندیاری، «راهکاری نوین جهت تولید دنباله آزمون کمینه در فرآیند آزمون نرم افزار با ترکیب الگوریتم‌های جستجوی تپه نوردی و جستجوی خفاش»، *مجله مهندسی برق*، دوره ۴۶، شماره ۳، ۳۵–۲۵، دانشگاه تبریز، پاییز ۱۳۹۵.
- [8] R. Premraj and K. Herzig, "Network Versus Code Metrics to Predict Defects: A Replication Study," *2011 Int. Symp. Empir. Softw. Eng. Meas.*, pp. 215–224, 2011.
- [9] Y. Zhou, B. Xu, and H. Leung, "On the ability of complexity metrics to predict fault-prone classes in object-oriented systems," *J. Syst. Softw.*, vol. 83, no. 4, pp. 660–674, 2010.
- [10] A. Shi, A. Gyori, M. Gligoric, A. Zaytsev, and D. Marinov, "Balancing trade-offs in test-suite reduction," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 246–256, 2014.
- [11] A. Shi, T. Yung, A. Gyori, and D. Marinov, "Comparing and Combining Test-suite Reduction and Regression Test Selection," *Proc. 2015 10th Jt. Meet. Found. Softw. Eng.*, pp. 237–247, 2015.
- [12] A. Gyori, A. Shi, F. Hariri, and D. Marinov, "Reliable testing: Detecting state-polluting tests to prevent test dependency," in *Proceedings of the 2015 International Symposium on Software Testing and Analysis*, pp. 223–233, 2015.
- [13] S. Eghbali and L. Tahvildari, "Test Case Prioritization Using Lexicographical Ordering," *IEEE Trans. Softw. Eng.*, vol. 5589, no. January, pp. 1–1, 2016.
- [14] H. Do, "Recent Advances in Regression Testing Techniques," in *Advances in Computers*, Elsevier, pp. 1–25, 2016.
- [15] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold, "Test case prioritization: An empirical study," in *Software Maintenance, 1999.(ICSM'99) Proceedings. IEEE International Conference on*, pp. 179–188, 1999.
- [16] C. Kaner, "Improving the maintainability of automated test suites," *Softw. QA*, vol. 4, no. 4, pp. 1, 1997.
- [17] P. K. Chittimalli and M. J. Harrold, "Recomputing coverage information to assist regression testing," *IEEE Trans. Softw. Eng.*, vol. 35, no. 4, pp. 452–469, 2009.
- [18] H. Do, S. Mirarab, L. Tahvildari, and G. Rothermel, "The effects of time constraints on test case prioritization: A series of controlled experiments," *IEEE Trans. Softw. Eng.*, vol. 36, no. 5, pp. 593–617, 2010.
- [19] D. Hao, L. Zhang, L. Zang, Y. Wang, X. Wu, and T. Xie, "To Be Optimal Or Not in Test-Case Prioritization," *IEEE Trans. Softw. Eng.*, vol. 6, no. 1, pp. 1–20, 2015.
- [20] Inozemtseva, Laura, and Reid Holmes. "Coverage is not strongly correlated with test suite effectiveness." In *Proceedings of the 36th International Conference on Software Engineering*, pp. 435-445. ACM, 2014.
- [21] Le Goues, Claire, and Westley Weimer. "Measuring code quality to improve specification mining." *IEEE Transactions on Software Engineering* 38, no. 1, pp. 175-190, 2012.
- [22] N. Nagappan and T. Ball, "Using software dependencies and churn metrics to predict field failures: An empirical case study," in *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*, pp. 364–373, 2007.
- [23] R. Subramanyam and M. S. Krishnan, "Empirical analysis of ck metrics for object-oriented design complexity: Implications for software defects," *IEEE Trans. Softw. Eng.*, vol. 29, no. 4, pp. 297–310, 2003.
- [24] T. Gyimothy, R. Ferenc, and I. Siket, "Empirical validation of object-oriented metrics on open source software for

-
- | | |
|---|--|
| <p>¹ Regression testing</p> <p>² Test case</p> <p>³ Test case prioritization</p> <p>⁴ Coverage criteria</p> <p>⁵ Lu</p> <p>⁶ Data fusion</p> <p>⁷ Average Percentage of Fault Detection (APFD)</p> <p>⁸ Paradigm</p> <p>⁹ Fault proneness</p> <p>¹⁰ Instrumentation</p> | <p>¹¹ Wu</p> <p>¹² http://www.eclemma.org/jacoco/</p> <p>¹³ https://maven.apache.org/index.html</p> <p>¹⁴ https://scitools.com/features/</p> <p>¹⁵ http://pitest.org/</p> <p>¹⁶ https://www.ibm.com/products/spss-statistics</p> <p>¹⁷ https://github.com/</p> <p>¹⁸ Srikanth</p> <p>¹⁹ Choudhary</p> <p>²⁰ Chen</p> |
|---|--|