تسریع فرایند کشف خطا در آزمون رگرسیون نرم‌افزار برای محیط‌های با محدودیت زمان و منابع

یلدا فضل‌علیزاده[۱]، علیرضا خلیلیان[۲]، محمد عبداللهی ازگمی[۳] و سعید پارسا[۴]

[۱] دانشکده مهندسی کامپیوتر دانشگاه علم و صنعت ایران، ya_alizadeh@comp.iust.ac.ir

[۲] دانشکده مهندسی کامپیوتر دانشگاه علم و صنعت ایران، khalilian@comp.iust.ac.ir

[۳] دانشکده مهندسی کامپیوتر دانشگاه علم و صنعت ایران، azgomi@iust.ac.ir

[۴] دانشکده مهندسی کامپیوتر دانشگاه علم و صنعت ایران، parsa@iust.ac.ir

چکیده - در مرحله نگهداری نرم‌افزار، بعد از هر بار تغییر جزئی کد، لازم است آزمون‌های جدیدی طراحی شوند، که اجرای آنها، به همراه تمام موارد آزمون قبلی، برای اعتبارسنجی نسخه جدید نرم‌افزار ضروری است. اجرای مکرر حجم زیاد آزمون‌ها، هزینه هنگفتی را بر توسعه‌دهندگان نرم‌افزار تحمیل می‌کند و به‌دلیل محدودیت‌های زمانی و منابع، آزمون کامل، اغلب غیرعملی است. برای حل این مشکل، از اولویت‌دهی موارد آزمون استفاده می‌شود. مشکلِ اغلب فنون اولویت‌دهی موجود این است که سلسله‌وار بودن اجرای آزمون‌ها، تأثیرگذاری کارایی آزمون‌ها در هر گام، بر کارایی مراحل بعدی آزمون و نیز محدودیت زمان و منابع آزمون در محیط‌های واقعی اجرای آزمون را نادیده می‌گیرند. لذا به‌دلیل همین محدودیت‌های اجرا، می‌توان از "پیشینه اجراهای قبلی موارد آزمون" در اولویت‌دهی آنها در اجراهای بعدی استفاده کرد. در این مقاله روشی برای اولویت‌دهی پیشینه محور موارد آزمون بر پایه دو معیارِ سابقه کشف خطای موارد آزمون در اجراهای قبل و نیز بر اساس سالمندی موارد آزمون ارائه شده است. نتایج مطالعات تجربی و بررسی موردی انجام شده، حاکی از بهبود عملکرد در کشف خطا و پایداری بیشتر نتایج روش فوق نسبت به روش اولویت‌دهی تصادفی است.

کلید واژه- آزمون رگرسیون نرم‌افزاری، اولویت دهی موارد آزمون، اولویت‌دهی پیشینه محور، کارایی تاریخی کشف خطا، سالمندی موارد آزمون.

۱- مقدمه

آزمون فرایندی است، برای آشکارسازی عیوب نرم‌افزار و مشخص کردن این‌که آیا نرم‌افزار با توجه به یکسری خصوصیات منتخب به سطح مشخصی از کیفیت رسیده است یا خیر [۱]. بسیاری از هزینه‌های توسعه نرم‌افزار، مربوط به آزمون‌های مکرر در "مرحله نگهداری نرم‌افزار در حال تکامل و اصلاح" است. زیرا ایجاد هر تغییر، حذف یا اضافه در کد نرم‌افزار، نسخه جدیدی از آن را بدست می‌دهد که نیازمند طراحی آزمون‌های جدید در هرگام و انجام مجدد تمام آزمون‌های قبل است. این آزمون‌های مکرر، که برای اطمینان از عدم تأثیرپذیری نامطلوبِ بخش‌های اصلاح نشده کد از تغییرات صورت می‌گیرد، "آزمون رگرسیون نرم‌افزار" نامیده می‌شود. به دلیل محدودیت زمان و منابعِ در دسترس برای گروه آزمون در دنیای واقعی، آزمون کامل اغلب غیرعملی است [۲،۱]. فنون گوناگونی برای حل مشکل هزینه زیاد آزمون رگرسیون ارائه شده است که یک از آنها "اولویت‌دهی موارد آزمون" است [۲]. اولویت‌دهی موارد آزمون به این معناست که بر اساس یک معیار، برای دستیابی *سریع‌تر* به هدف آزمون، ترتیب‌دهی خاصی بر اجرای دنباله آزمون‌ها اعمال شود. یکی از اهداف اصلی از اولویت‌دهی موارد آزمون، یافتن بهترین ترتیب اجرایِ دنباله آزمون‌ها، برای "افزایش نرخ کشف خطا تا حد امکان" است [۳]. از طرفی، در محیط‌هایی با محدودیت زمان و منابع آزمون، نیاز شدیدتری به اعمال فنونی جهت " پیش‌تر انجام دادن مهمترین آزمون‌ها" وجود دارد. مسئله یافتن بهترین ترتیب اجرای ممکن، معادل با مسئله کوله‌پشتی 0/1 است که NP-hard بوده و بدون راه‌حل قطعی است [۲]. در اغلب فنون اولویت‌دهی موجود، دید محدود شده‌ای وجود دارد و محققان در مطالعات تجربیشان عموماً آزمون رگرسیون را به عنوان یک آزمون "یک دفعه‌ای" در نظر گرفته‌اند؛ درحالی‌که آزمون رگرسیون، ذاتاً آزمونی "ادامه‌دار" و "دراز مدت" است و بهترین

مدل برای آن، اجراهای سلسله‌وار آزمون‌ها بعد از هر تغییر نرم‌افزار است؛ به‌گونه‌ای که کارایی هر کدام از آنها بر آزمون‌های بعدی مؤثر است.

در همین راستا، در [۴] نخستین بار بحث "حافظه‌دار کردن" آزمون رگرسیون و استفاده از اطلاعات پیشینه کارایی موارد آزمون، در اولویت‌دهی مجموعه آزمون مطرح گردید. فن مکاشفه‌ای ارائه شده در [۴] در واقع ترکیبی است از "فن انتخابِ مبتنی بر سابقه در آزمون رگرسیون" و سپس اولویت‌دهی موارد آزمون. به این ترتیب که در هر گام از پیشینه اجرای موارد آزمون، برای انتخابهای بعدی موارد آزمون استفاده می‌شود و در هر مرحله زیر مجموعه‌ای از مجموعه آزمون اولیه (بدون کاهش دائمی آن) را برای اولویت دهی و اجرا بر روی نسخه جدید نرم‌افزار انتخاب می‌کند. نشان داده شده است [۴] که چنین مکاشفه‌ای می‌تواند با گذشت اجراهای طولانی، هزینه را کاهش داده و کارایی آزمون رگرسیون را در محیط‌های توسعه محدودیت‌دار کاهش دهد.

مشکل فنِ پیشینه محورِ جاری در این است که در اطلاعات پیشینه اجرای موارد آزمون، تنها اثر *اخیراً* اجرا شدن یا نشدن و یا آشکارسازی خطا یا عدم آن، *در اجرای قبل، به صورت صفر و یک* در رابطه بازگشتیِ احتمال انتخاب موارد آزمون در نظر گرفته شده است. در حالی‌که برای اولویت‌دهی کارای پیشینه‌محور، هم کارایی کشف خطا در طول اجراها باید در نظر گرفته شود و هم از *اولویت‌دهی پایین و منسوخ شدن دائمی یک‌سری از آزمون‌ها* بایستی جلوگیری شود. در روش پیشنهادی در این مقاله، به ارائه راه‌حلی برای اولویت‌دهی پیشینه محور موارد آزمون در هر گام اجرا، بر اساس سابقه کارایی موارد آزمون در کشف خطا و نیز سالمندی موارد آزمون و ارزیابی این روش بر روی برنامه محک زیمنس پرداخته می‌شود. در ادامه مقاله در بخش ۲، تعاریف مختلف پیشینه اجرای آزمون و اولویت‌دهی پیشینه‌محور آمده است. پس از بیان مشکل روش موجود، در بخش ۳، روش پیشنهادی اولویت‌دهی پیشینه محور شرح داده شده است. در بخش ۴، متریک ارزیابی و برنامه‌های محک مورد استفاده در مطالعه تجربی و بررسی موردی، معرفی شده‌اند و نتایج ارزیابی بر روی برنامه‌ها شرح داده شده است. بخش ۵ به نتیجه‌گیری اختصاص دارد.

۲- اولویت‌دهی پیشینه محور موارد آزمون

نخستین بار در سال ۱۹۹۷ مسئله اولویت‌دهی موارد آزمون مطرح گردید [۵] و پس از آن در سالهای بعد، اولویت‌دهی موارد آزمون توسط رودرمل، اِلباوم و چند تن دیگر از محققان این حوزه ادامه یافت [۳،۶،۷،۸،۹،۱۰،۱۱]. بیشتر فنون اولویت‌دهی موجود، کد محور و بر پایه روشهای حریصانه‌اند. انواع دیگر، فنون اولویت‌دهی مدل محور [۱۲] و اولویت‌دهی پیشینه محور [۴،۱۳] هستند.

تعریف اولویت‌دهی پیشینه محور موارد آزمون [۴] این‌گونه است: فرض کنیم احتمال انتخاب هر مورد آزمون tc در زمان t، $P_{tc,t}(H_{tc},\alpha)$ در نظر گرفته شود، به طوری که H_{tc}، مجموعه‌ای از مشاهدات مرتب شده بر اساس زمان، به صورت $\{h_1, h_2, h_3, ..., h_t\}$ باشد، که از اجراهای قبلی برای هر مورد آزمون بدست آمده‌اند و نشانگر عملکرد مورد آزمون tc در طول پیشینه اجرای آن هستند. بر این اساس، احتمال انتخاب هر مورد آزمون در مجموعه T' بر اساس پیشینه اجرای آن به‌صورت روابط زیر قابل تعریف است:

$$P_k = \alpha h_k + (1-\alpha) P_{k-1} \qquad (۱)$$

$$P_0 = h_1 \qquad k \geq 1, 0 \leq \alpha < 1$$

در روابط فوق α یک ثابت هموار کننده است که برای وزن‌دهی به تاریخچه مشاهدات به‌کار رفته است.

تعاریف متفاوتی که از H_{tc} در رابطه (۱) بیان می‌شود، نشانگر این است که اولویت‌دهی پیشینه محور می‌تواند بر پایه ملاک‌های گوناگونی انجام شود. برخی از این تعاریف عبارتند از:

۱) تاریخچه اجرا. برای هر جلسه آزمون i که مورد آزمون tc در آن اجرا شود، h_i در احتمال انتخاب بعدی مقدار 0 و در غیر این صورت 1 می‌گیرد. به عبارت دیگر، موارد آزمون با اولویت بالاتر، پس از اجرا شدن در یک جلسه، در جلسات بعدی احتمال انتخاب پایین‌تری نسبت به قبل خواهند داشت. به این ترتیب در محیط‌های محدودیت‌دار، در خلال چندین جلسه آزمون در بین تمام موارد آزمون گردش می‌شود. مقدار α کوچکتر، احتمال انتخاب بزرگتری به موارد آزمونی که اخیراً اجرا نشده‌اند تخصیص می‌دهد. مشکل این تعریف آن است که با اجرای موارد آزمون کاراتر، احتمال انتخاب آنها نیز همانند بقیه موارد آزمون تضعیف می‌شود و کاراتر بودن موارد آزمون هیچ تأثیری در بالا رفتن اولویت و تسریع انتخاب‌های بعدی آنها ندارد.

۲) کارایی کشف خطای نشان داده شده. برای هر جلسه آزمون i، که مورد آزمون tc در آن خطا آشکار کند، h_i مقدار 0 و در غیر این صورت 1 می‌گیرد. هرگاه مقدار α بزرگ باشد احتمال بالاتر به موارد آزمونی تخصیص داده می‌شود که در اجرای اخیر، خطا آشکار کرده‌اند. اثر این تعریف برای H_{tc} این

است که اجرای موارد آزمونی را کـه بنـدرت و یـا اصـلاً خطـایی آشکار نکرده‌اند، محدود می‌کند.

۳) پوشش مؤلفه‌های برنامه. منظـور از مؤلفـه‌هـای برنامـه، جملات، مسیرها، توابـع، زوج‌هـای تعریـف-اسـتفاده و ماننـد آن است. در این تعریف از H_{tc}، به عنـوان مثـال، اولویـت بـالاتر بـه موارد آزمونی داده می‌شود که توابعی را می‌پوشانند که با فرکانس کمتری اخیراً پوشش یافته‌اند (اجرا شده‌اند).

فن پیشنهاد شده در (۱) در حقیقت یک "انتخـاب پیشـینه محور" در آزمون رگرسیون نرم‌افزاری است و لازمه اولویـت‌دهـی موارد آزمون این است که از یکی از فنون موجود اولویت‌دهـی بـر روی نتایج این انتخاب استفاده شود (همانند [۱۳]، که با ترکیـب دو روش انتخاب مبتنی بر پیشینه و فن اولویت‌دهی هزینه-آگـاه [۱۰]، فنی برای اولویت‌دهی پیشینه-محورِ آگـاه از هزینـه ارائـه داده است).

مشکل روش پیشینه محور موجـود ایـن اسـت کـه انتخـاب پارامتر h_k (مجموعه مشاهدات تاریخی اجراهای هر مورد آزمـون از اولین تا k اُمین اجرا) که تنها با دو مقدار0 یا 1، آن هم فقط بر اساس اجرای مرحله قبل مورد آزمون، در تعیین احتمال انتخـاب جاری هر مورد آزمون شرکت می‌کنـد، مـلاک مناسـبی نیسـت. به‌علاوه این‌که مورد آزمون اخیراً اجرا نشده باشد و یا ایـن‌کـه در اجرای قبلی خطا آشکار کرده یا خیر، ملاک مناسـبی بـرای بـالا رفتن احتمال انتخاب آن در گام اجرای بعدی نیست.

۳- روش پیشنهادی

در افزایش اولویت یک مورد آزمون در انتخابهای بعدی، چند عامل مؤثر است که یکی از آن‌هـا، کـارایی کشـف خطـای مـورد آزمون در تعداد دفعاتی است که اجرا شده است. مثلاً اگـر مـورد آزمون tc_A، ۳ خطا در۲۰ بار اجرا و مورد آزمون tc_B، ۹ خطـا در ۱۸ بار اجرا آشکار کرده باشند، کارایی تـاریخی نشـان مـی‌دهـد مورد آزمون tc_B بیش از ٪۳۵ در کشف خطا نسبت به tc_A بهتـر عمل کرده است و اولویت بالاتری باید به آن داده شود. مثال فوق نشان می‌دهد که تعداد اجرای موارد آزمون و تعـداد دفعـاتی کـه سبب آشکار شدن وجود خطا در نرم‌افزار شده‌اند، توأم با هم، یک عامل تأثیرگذار در پیشینه کارایی موارد آزمون هستند. بنـابراین، اگر در k اُمین اجرای آزمون رگرسیون، تعداد دفعاتی کـه اجـرای مورد آزمون tc، بر روی نسخه جدید نرم‌افزار، بـا شکسـت مواجـه شده باشد (اجرای آن وجود یک یا چند خطا را در نرم‌افزار اصلاح شده نشان دهد) را با fc_k نشان دهیم و کل تعداد دفعات اجـرای آن تـا ایـن مرحلـه را نیـز بـا ec_k نشـان دهـیم (در محـیط محدودیت‌دار تمام موارد آزمون در هر مرتبـه اجـرا نمـی‌شـوند)، برای هر مورد آزمون، می‌توان رابطه اولویت مورد آزمون با کارایی آن در k اُمین اجرا را به‌صورت رابطه (۲) نوشت:

$$PR_k \approx \frac{fc_k}{ec_k} \qquad (۲)$$

$$fc_k = \sum_{i=1}^{k-1} f_i \qquad (۳)$$

$$ec_k = \sum_{i=1}^{k-1} e_i \qquad (۴)$$

$$f_i = \begin{cases} 1 & \text{اگر مورد آزمون در جلسه آزمون } i \text{ اُم خطا آشکار کند} \\ 0 & \text{در غیر این صورت} \end{cases}$$

$$e_i = \begin{cases} 1 & \text{اگر مورد آزمون در جلسه آزمون } i \text{ اُم اجرا شده باشد} \\ 0 & \text{در غیر این صورت} \end{cases}$$

به عبارت دیگر، اولویت پیشینه محور هر مورد آزمون در هـر گام، با نسبت آشکارسازی خطا بر کـل دفعـات اجـرای آن مـورد آزمون، متناسب است.

عامل مؤثر دیگر بر اولویت‌دهی این است کـه مـوارد آزمـون برای مدت طولانی اجرا نشده باقی نماننـد. بـه عبـارت دیگـر، در جریان تکرارهای آزمون رگرسیون، باید اطمینان داشته باشیم که هر مورد آزمون بالاخره اجرا شده و خطاهای قابل کشـف توسـط آن، آشکار خواهند شد. در همین راستا، عامل h_k در k اُمین اجرا برای هر مورد آزمون به صورت رابطه (۵) تعریف می‌شود:

$$h_k = \begin{cases} 0 & \text{اگر مورد آزمون در جلسه آزمون } k\text{-1 اُم اجرا شده باشد} \\ h_{k-1}+1 & \text{در غیر این صورت} \end{cases} \qquad (۵)$$

$$h_0 = 0$$

در حقیقت عامل h_k مانند یک "شمارنده" عمل می‌کند. در سیستم عامل و مفاهیم مربوط به زمانبندی پردازه، مشکلی به‌نام "قحطی‌زدگی" وجود دارد؛ به این معنی که اگر در جریان اجـرای پردازش‌های متعددِ موجود در سیستم، نوبـت اجـرا بـرای مـدت طولانی به پردازشی تعلق نگیرد، تعداد این دفعات به‌عنوان "سن" آن در نظـر گرفتـه مـی‌شـود و بـرای جلـوگیری از سـالمندی پردازش‌ها، شمارنده‌ای برای سن هـر کـدام تخصـیص مـی‌یابـد. شـمارنده h_k نیـز، نظیـر همـین نقـش را بـرای دخالـت دادن سالمندی هر مورد آزمون در اولویت‌دهی آن دارد. لـذا مـی‌تـوان نوشت:

$$PR_k \approx h_k \qquad (۶)$$

به عبارتی شمارنده (سن) هر مورد آزمون و در نتیجه اولویت اجرای آن، در هر بار اجرا نشدن افزایش می‌یابد، تا هنگـامی کـه

جزء موارد آزمون اجرایی قرار بگیرد و اجرا شود. در صورت اجرای مورد آزمون، شمارنده (سن این مورد آزمون) برابر 0 می‌شود و همین عملیات تکرار می‌شود.

در نهایت، سومین عاملی که باید در اولویت دهی موارد آزمون در هر گام اجرا دخالت داشته باشد [۴]، "اولویت و رتبه پیشین هر مورد آزمون در اجرای قبلی" آزمون رگرسیون است. به عبارت دیگر:

$$PR_k \approx PR_{k-1} \qquad (۷)$$

استفاده از اولویت اجرای قبلی مورد آزمون و تعریف رابطه به‌صورت بازگشتی، اولاً باعث "نرمتر شدن انتخاب موارد آزمون" و عدم بروز تغییرات شدید در رشته آزمون اجرایی در هر گام اجرا، نسبت به گام قبلی می‌شود؛ ثانیاً در مواردی که از لحاظ نسبت تعداد دفعات آشکارسازی خطا بر اجرا و نیز سالمندی وضعیت یکسانی وجود داشته باشند، باید عامل دیگری در تعیین اولویت صحیح بین آنها قائل شد. این مهم با استفاده از اولویت اجرای قبلی موارد آزمون تأمین می‌شود. در رابطه پیشنهادی، مقدار PR_0 را می‌توان برحسب معیارهای گوناگون، مثلاً درصد پوشش کدِ موارد آزمون محاسبه کرد؛ که به این ترتیب، اثر پوشش موارد آزمون، به‌دلیل بازگشتی بودن رابطه، در تمامی‌اولویت‌دهی‌های بعدی هم شرکت می‌کند؛ که شاخص بسیار مناسبی برای قدرت موارد آزمون در تمام فنون اولویت‌دهی کد محور محسوب می‌شود. زیرا منطقی است که فرض کنیم موارد آزمونی که درصد بیشتری از کد برنامه را می‌پوشانند (اجرا می‌کنند)، احتمال پوشش دادن قسمت‌های خطادار و فعال کردن و آشکارسازی خطاها در آنها بیشتر است، لذا اولویت بالاتری به آنها داده می‌شود. به این ترتیب، رابطه اولویت‌دهی پیشنهادی، در حقیقت یک فن اولویت‌دهی است که اساس آن، پیشینه کارایی موارد آزمون در کشف خطا بوده و نیز پوشش محور محسوب می‌شود.

بنابر آن چه در روابط (۵)، (۶) و (۷) گفته شد، می‌توان رابطه اولویت‌دهی در k اُمین گام اجرا برای هر مورد آزمون را به صورت زیر نوشت:

$$PR_k = \alpha \frac{fc_k}{ec_k} + (1-\alpha) PR_{k-1} + (1-\alpha)^2 h_k \qquad (۸)$$

$$0 \le \alpha < 1,\ k \ge 1$$

با تغییر ثابت هموارکننده α در رابطه فوق، می‌توان به کنترل میزان تأثیر عوامل مؤثر در اولویت‌دهی پرداخت. به‌علاوه ضریب $(1-\alpha)^2$ به عنوان ضریب h_k در رابطه اولویت، که نسبت به ضرایب α و $1-\alpha$ تا حدی کوچک‌کننده‌تر است، باعث می‌شود میزان تأثیر سالمندی، که بر حسب شرایط اجرای مورد آزمون، هر مرتبه یک واحد افزایش خواهد داشت، در مقابل مقادیر ec_k / fc_k و PR_{k-1} که هر دو مقادیری کسری هستند، به اشتباه تأثیر بسیار بالایی در اولویت‌دهی نداشته باشد و تأثیر عوامل فوق را نپوشاند.

در رابطه (۱) آنچه برای P_k بدست می‌آید، احتمال انتخاب مورد آزمون است و پس از انتخاب زیر مجموعه 'T از کل مجموعه آزمون T، باید از یک فن اولویت‌دهی استفاده کرد. در حالی‌که در رابطه (۸)، PR_k اولویت مورد آزمون را در k اُمین گام اجرا مشخص می‌کند. پس از اولویت‌دهی، با توجه به زمان و منابع محدود آزمون، تعداد کافی از موارد آزمون با شروع از اولویت‌های بالا اجرا می‌گردد.

۴- ارزیابی روش پیشنهادی

۴-۱- متریک ارزیابی

برای ارزیابی سرعت کشف خطای فنون اولویت دهی موارد آزمون، از متریک APFD که "متوسط وزن‌دارِ درصد خطاهای کشف شده در اجرای مجموعه آزمون" است، استفاده می‌شود [۲]. محاسبه APFD در اصل به صورت رابطه (۹) است:

$$APFD = 1 - \frac{TF_1 + TF_2 + \ldots + TF_m}{nm} + \frac{1}{2n} \qquad (۹)$$

در رابطه (۹)، n تعداد موارد آزمون و m تعداد خطاهای موجود را نشان می‌دهد. هر TF_i در رابطه فوق، محل مورد آزمونی را در دنباله مرتب موارد آزمون در رشته اولویت‌دهی شده نشان می‌دهد، که برای اولین بار خطای i اُم را آشکار نموده است. هرچه مقدار APFD محاسبه شده، به ٪۱۰۰ نزدیک‌تر باشد، کارایی آن فن اولویت‌دهی در کشف سریع‌تر خطاها بیشتر است.

۴-۲- مطالعه تجربی و بررسی موردی

استفاده از مجموعه زیمنس [۱۴] که شامل هفت برنامه به زبان C است، در مطالعات تجربی برای ارزیابی فنون اولویت‌دهی موارد آزمون متداول است. مجموعه زیمنس شامل برنامه، زیرمجموعه‌هایی از موارد آزمون برای هر برنامه و اسکریپت‌هایی برای اجرای برنامه است. همچنین، زیرمجموعه‌هایی تصادفی از موارد آزمون (با تعداد برابرِ مورد آزمون با زیرمجموعه‌های متناظر با پوشش انشعاب)، جهت مقایسه فنون اولویت‌دهی با اجرای تصادفی موارد آزمون ایجاد شده‌است. برای هر برنامه، تعدادی نسخ تک‌خطایی هم ایجاد شده است، که در هر نسخه، خطاها به

طور دستی توسط تیم‌هایی مجزا کاشته شده است. در ارزیابی روشهای اولویت‌دهی، به نسخ چندخطایی نیاز است. به همین دلیل، مجموعه‌ای ترکیبی از تک‌خطاهای دو به‌دو مجزا ایجاد شده است. این خطاها، بنابرتجارب برنامه‌نویسان از انواع خطاهای متداول ایجاد شده و به خطاهای واقعی بسیار نزدیک هستند.

بررسی موردی انجام شده در این مقاله، بر روی برنامه محک space صورت گرفته است، که برنامه‌ای بزرگ (10KLOC) و با خطاهای واقعی است [۶] و نتایج اولویت‌دهی روش پیشنهادی را بر روی مجموعه آزمون این برنامه محک، با روش اولویت‌دهی تصادفی موارد آزمون مقایسه می‌کند.

در مطالعه تجربی و بررسی موردی انجام شده، از مجموعه‌های چندخطایی، ۳۰ نسخه چندخطایی به تصادف انتخاب و به عنوان ۳۰ مرحله از تکرار آزمون رگرسیون و برای "بررسی عملکرد روش پیشنهادی در طول پیشینه اجرا"، از آنها استفاده شده است. در هر کدام از ۳۰ نسخه، روش اولویت‌دهی پیشنهادی روی ۱۰۰۰ مجموعه با پوشش انشعاب اجرا شده است. در پایان هر گام اجرا و پس از کشف خطاها، متریک APFD برای رشته اولویت‌دهی شده موارد آزمون محاسبه شده است. در پیاده‌سازی روش پیشنهادی، به‌دلیل محدودیت در منابع آزمون، تنها درصدی (مثلاً تنها یک‌دهم) از کل مجموعه اولویت‌دهی شده اجرا می‌شود. لذا برخلاف روشهای اولویت‌دهی موجود، که به اجرای کل زیرمجموعه اولویت‌دهی شده می‌پردازند، روش پیشنهادی با اجرای تعداد کمترِ موارد آزمون، در واقع به "صرفه‌جویی در منابع آزمون" نیز کمک کرده است.

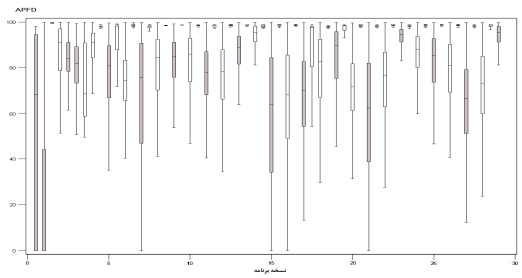
برای نمایش و مقایسه کارایی روش اولویت‌دهی پیشنهادی، از روش متداول در مطالعات تجربی، یعنی نمودار جعبه‌ای استفاده شده است که نتایج *متوسط کشف خطا برای ۱۰۰۰ دنباله آزمون با پوشش انشعاب با روش پیشنهادی* را، با *متوسط کشف خطای ۱۰۰۰ دنباله آزمون با اولویت تصادفی* مقایسه می‌کند. از آنجا که هر یک از ۱۰۰۰ دنباله آزمون، برای ۳۰ نسخه چندخطایی اولویت‌دهی شده و با دنباله تصادفی متناظر آن مقایسه می‌شود، برای هر یک از هشت برنامه، ۳۰ جفت جعبه مجزا رسم شده است. هر زوج، نتایج متوسط کشف خطا (مقدار APFD) توسط روش اولویت‌دهی پیشنهادی را، در جعبه سمت چپ و روش اولویت تصادفی را، در جعبه سمت راست نشان می‌دهد و در هشت نمودار (الف تا ح) در شکل ۲ نشان داده شده‌اند. نمودار جعبه‌ای [۱۵]، نموداری است که به کمک معیارهای مرکزی و پراکندگی، توزیع مجموعه داده‌ها را به شکلی گویا و مفید ارائه می‌دهد. این نمودار، با استفاده از یک مستطیل و دو خط در دو طرف مستطیل (ویسکر) برای چارکهای اول و سوم داده‌ها و نیز خط میانه داده‌ها در وسط جعبه رسم می‌شود. خطوط خارج شده از جعبه‌ها، حداقل و حداکثر داده‌هایی را که پرت و برون‌هشته نیستند، نشان می‌دهد.

در نمودار جعبه‌ای، گستره جعبه‌ها در امتداد محور عمودی (گستره مقادیرAPFD)، پراکندگی مقادیر به‌دست آمده برای متوسط کشف خطا را نشان می‌دهد. هر چه گستردگی جعبه کمتر باشد، به این معنی است که عملکرد روش اولویت‌دهی، پایداری بیشتری دارد و رفتار یکنواخت‌تری از نظر سرعت کشف خطا از خود نشان می‌دهد. روشن است که بالاتر بودن محل جعبه (بیشتر بودن مقادیر APFD)، حاکی از قدرت روش اولویت‌دهی در کشف سریع‌تر خطا است. همان‌طور که در شکل ۲ ملاحظه می‌شود، روش پیشنهادی، تا حد قابل ملاحظه‌ای هم از لحاظ کشف سریع‌تر خطاها (بالاتر بودن جعبه) و هم از لحاظ پایداری عملکرد در کشف خطا، در اولویت‌دهی دنباله‌های گوناگون (گستردگی کمتر جعبه)، بهتر عمل کرده است.

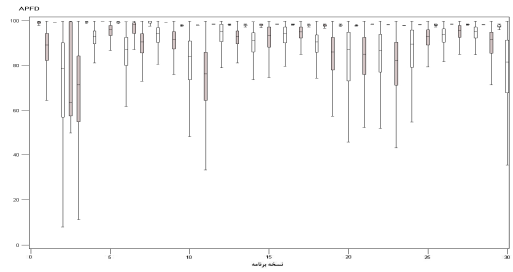
۵- نتیجه‌گیری

در این مقاله، روش جدیدی برای اولویت‌دهی مبتنی بر پیشینه موارد آزمون ارائه شد؛ که با در نظرگیری محدودیت‌های واقعی محیط اجرا و هزینه زیاد اجرای تکراری کل موارد آزمون، تنها بخشی از دنباله آزمون اولویت‌دهی شده را اجرا می‌کند.

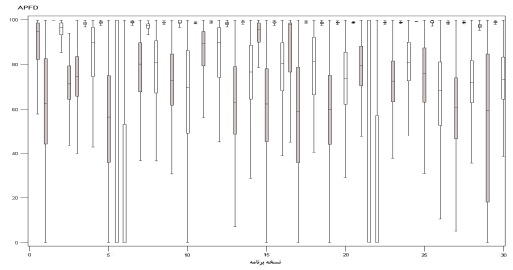
در روش پیشنهادی، بالا بودن احتمال اجرای مورد آزمون، سابقه کارایی آن در جریان اجرای طولانی مدت در کشف خطا و مدت طولانی اجرا نشدن (سالمندی) مورد آزمون، سبب بالا رفتن اولویت اجرای آن می‌شود. این روش، مستقیماً سه عامل فوق را در اولویت‌دهی دخالت می‌دهد و از این لحاظ با روش‌های موجود، که ترکیبی از دو مرحله انتخاب پیشینه محور و سپس استفاده از فنون اولویت‌دهی موجود است، تفاوت عمده دارد.



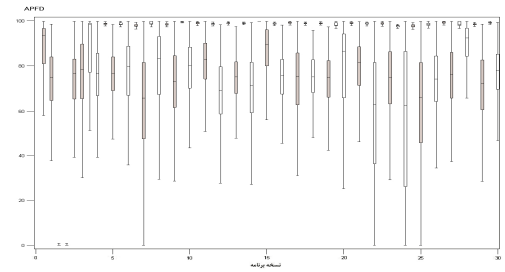
الف) مطالعه تجربی: برنامه print_tokens



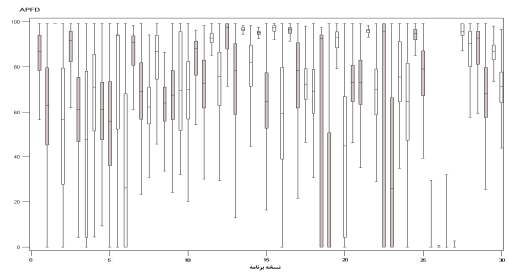
ب) مطالعه تجربی: برنامه print_tokens2



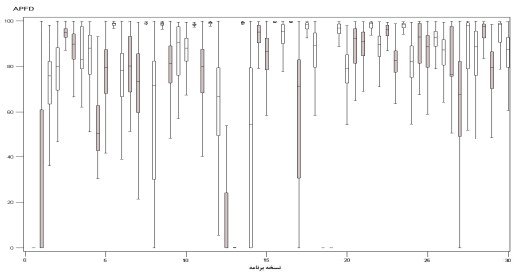
ج) مطالعه تجربی: برنامه schedule



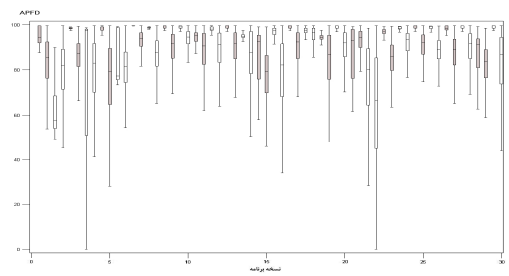
د) مطالعه تجربی: برنامه schedule2



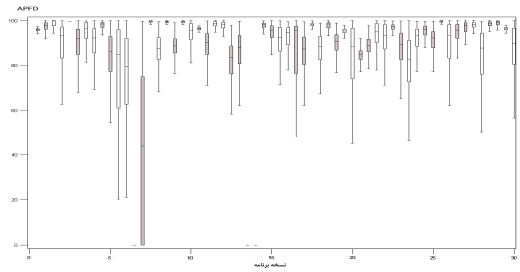
ه) مطالعه تجربی: برنامه tcas



و) مطالعه تجربی: برنامه replace



ز) مطالعه تجربی: برنامه tot-info



ح) بررسی موردی: برنامه space

شکل ۲) نمودار جعبه‌ای مقایسه روش پیشنهادی با اولویت تصادفی.

مراجع

[1] I. Burnstein, *Practical software testing: a process-oriented approach.* ISBN 0-387-95131-8, Springer-Verlag New York, Inc, 2003.

[2] G. Rothermel, R. H. Untch, C. Chu and M. J. Harrold, "Test case prioritization: an empirical study," *Proc. IEEE Int. Conf. on Software Maintenanc.* Oxford, England, pp. 179-188, 1999.

[3] G. Rothermel, R. H. Untch, C. Chu and M. J. Harrold, "Prioritizing Test Cases for Regression Testing," *Proc. IEEE Transactions on Software Engineering*. pp. 102-112, 2001.

[4] J. M. Kim and A. Porter, "A History-Based Test Prioritization Technique for Regression Testing in Resource Constrained Environment," *Proc. 24th Int'l Conf. Software Engineering*. pp. 119-129, 2002.

[5] W. E. Wong, J. R. Horgan, S. London and H. Agrawal, "A study of effective regression testing in practice," *Proc. 8th Int'l Symp. Software Reliability Engineering.* pp. 230-238, 1997.

[6] S. Elbaum, A. G. Malishevsky and G. Rothermel, "Test Case Prioritization: A Family of Empirical Studies," *Proc. IEEE Transactions on Software Engineering,* Vol. 28, No. 2, pp. 159-182, 2002.

[7] S. Elbaum, A. G. Malishevsky and G. Rothermel, "Prioritizing test cases for regression testing." *Proc. 7th Int'l Syrup. Software Testing and Analysis*. pp. 102-112, Aug. 2000.

[8] Z. Li, M. Harman, and R. M. Hierons, "Search Algorithms for Regression Test Case Prioritization," *Proc. IEEE Transactions on Software Engineering,* Vol. 33, pp. 225-237, 2007.

[9] S. Elbaum, G. Rothermel, S. Kanduri, and A. G. Malishevsky, "Selecting a Cost-Effective Test Case Prioritization Technique," *Software Quality Control,* Vol. 12, pp. 185-210, 2004.

[10] A. G. Malishevsky, J. R. Ruthruff, G. Rothermel and S. Elbaum, "Cost-cognizant test case prioritization," Department of Computer Science and Engineering, Nebraska, Lincoln, Tech. Rep. TR-UNL-CSE-2006-0004 March 2006.

[11] P. R. Srivastava, "Test Case prioritization," JATIT, Computer Science and Information System Group, BITS Pilani, India-333031, 2005 – 2008.

[12] B. Korel, G., Koutsogiannakis and L. H. Tahat, "Model-based test prioritization heuristic methods and their evaluation," *Proc. 3rd Int'l Workshop on Advances in Model-Based Testing,* London, UK, July 2007.

[13] H. Park, H. Ryu and J. Baik, "Historical Value-Based Approach for Cost-Cognizant Test Case Prioritization to Improve the Effectiveness of Regression Testing," *paper appears in: 2th Int'l Conf. Secure System Integration and Reliability Improvement,* pp. 39-46, Yokohama, Japan, 2008.

[14] G. Rothermel, S. Elbaum, A. Kinneer and H. Do, Software-artifact infrastructure repository, http://www.cse.unl.edu/~gal ileo/sir.

[15] http://en.wikipedia.org/wiki/Box_plot.