رویکردی جدید جهت تغییر ترتیب اجرای موارد آزمون در آزمون رگرسیون نرم‌افزار در محیط‌های دارای محدودیت منابع

یلدا فضل علیزاده[۱]، علیرضا خلیلیان[۲]، محمد عبداللهی ازگمی[۳]، سعید پارسا[۴]

[۱] دانشکده مهندسی کامپیوتر دانشگاه علم و صنعت ایران
تهران، 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
تهران، ایران
azgomi@ just.ac.ir

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

چکیده

[illegible]

چکیده

بسیاری از هزینه های توسعه نرم‌افزار، مربوط به آزمون‌های مکرر، در «مرحله نگهداری نرم‌افزار در حال تکامل و اصلاح» است. زیرا ایجاد هر تغییر جزئی در کد نرم‌افزار، نسخه جدیدی از آن را بدست می‌دهد که اعتبارسنجی آن، نیازمند طراحی و اجرای آزمون‌های جدید در هر گام و انجام مجدد تمام آزمون‌های قبل است. به این ترتیب اطمینان حاصل می‌شود که تغییرات صورت گرفته، عملکرد جاری نرم‌افزار را به صورت نامطلوب تحت‌تأثیر قرار نداده است. این فرایند «آزمون رگرسیون نرم‌افزار» نامیده می شود. بدلیل محدودیت منابع و زمان، ممکن نیست کل این حجم زیاد آزمون‌های رو به توسعه را در هر دور تکرار آزمون، مجددا اجرا کرد. برای غلبه بر این مشکل، لازم است اجرای آزمون‌ها «اولویت‌دهی» شود، به نحوی که اجرای مهمترین موارد آزمون، برحسب هدف در نظر گرفته شده برای آزمون، زودتر انجام شود. راهکارهای مختلفی برای اولویت‌دهی موارد آزمون وجود دارد که یکی از آن‌ها، استفاده از پیشینه اجراهای قبلی موارد آزمون است. در این مقاله روشی برای اولویت‌دهی پیشینه محور موارد آزمون بر پایه دو معیار سابقه کشف خطا و سالمندی موارد آزمون ارائه شده است. مطالعات تجربی انجام شده، بهبود و پایداری بیشتر نتایج روش فوق را نسبت به اولویت‌دهی تصادفی تأیید می‌کند.

کلمات کلیدی

آزمون رگرسیون نرم‌افزار[**]**، اولویت دهی موارد آزمون**[**]**، اولویت‌دهی پیشینه محور**[**]**، کارایی تاریخی کشف خطا**[**]**، سالمندی موارد آزمون.**

۱- مقدمه

در جریان توسعه و نگهداری نرم‌افزار برای رفع خطاهای موجود در نرم‌افزار و نیز انعکاس و پیاده‌سازی تغییراتی که در خصوصیات برنامه ایجاد شده است، مکرراً نرم‌افزار اصلاح می‌شود و تغییراتی در آن داده می‌شود. پس از هر تغییر در نرم‌افزار، برای بررسی اینکه رفتار آن به‌جز قسمت‌های تغییر یافته، بلا تغییر مانده است و نیز برای اعتبارسنجی مجدد پوششی که مجموعه آزمون با توجه به یک معیار پوشش خاص برای نرم‌افزار، مثل دستورالعمل ایجاد می‌کرد، تکرار آزمون‌های فعلی ضرورت دارد. در عین حال اعتبارسنجی قسمتهای تغییر یافته نرم افزار هر بار نیازمند طراحی تعدادی آزمون‌های جدید است که به این ترتیب حجم آزمون‌های نرم‌افزار مرتباً رو به افزایش است. بنابراین این فاز ضروری در توسعه محصول نرم‌افزاری که برای آشکارسازی عیوب نرم‌افزار و مشخص کردن سطح کیفی آن با توجه به یکسری خصوصیات منتخب انجام می‌شود[۱]، بسیاری از هزینه‌های توسعه نرم‌افزار (بین ۳۰ تا ۵۰ درصد) را به خود اختصاص می‌دهد. این آزمون‌های مکرر در"مرحله نگهداری نرم افزار در حال تکامل و اصلاح" که با ایجاد هر تغییر در کد نرم‌افزار، برای اطمینان از عدم تأثیر پذیری نامطلوب بخشهای اصلاح نشده کد از تغییرات صورت می‌گیرد " آزمون رگرسیون نرم‌افزار" نامیده می‌شود.

به دلیل محدودیت زمان و منابع در دسترس برای گروه آزمون در دنیای واقعی، آزمون کامل اغلب غیر عملی است[۲،۱]. فنون گوناگونی برای حل مشکل هزینه زیاد آزمون رگرسیون ارائه شده است. در بین این فنون، سه فن عمده که بر پایه استفاده مجدد از مخزن آزمون استوارند عبارتند از: "انتخاب موارد آزمون در آزمون رگرسیون نرم‌افزار"، "کاهش دنباله آزمون" و "اولویت‌دهی موارد آزمون" [۲]. فن اولویت‌دهی موارد آزمون تلاش می‌کند با ترتیب‌دهی اجرای موارد آزمون بر اساس یک معیار شایستگی[۱] به بهبود آزمون رگرسیون نرم افزاری بپردازد بگونه ای که مهمترین ها ابتدا اجرا شوند. یکی از اهداف عمده[۳] در اولویت‌دهی موارد آزمون این است که با اجرای موارد آزمونی که تعداد بیشتری از خطاها را اجرا می کنند، متوسط کشف خطا را در طول اجرای آزمون تا جای ممکن افزایش داد تا با کشف سریعتر این خطاها امکان بازخورد سریعتر به تیم توسعه نرم‌افزار و ارسال آن برای ویرایش و تصحیح خطا میسر شود. با این کار می توان مطمئن بود که در صورت توقف و ناتمام ماندن جریان آزمون در هرجا، مؤثرترین آزمونها انجام گرفته و حداکثر خطاهای موجود کشف شده اند.

مسئله اولویت‌دهی موارد آزمون در حقیقت تعیین جایگشتی از کلیه موارد آزمون موجود در رشته آزمون است، به گونه ای که این ترتیبِ اجرا، آشکار کننده حداکثر خطاهای ممکن باشد. بنابرین مسئله اولویت‌دهی موارد آزمون در حقیقت یک مسئله جستجو است که در مراجع این مسئله و پیچیدگی آن را معادل با مسئله کوله پشتی می‌دانند که NP-hard بوده و بدون راه حل قطعی است. لذا راه حلهای موجود برای مسئله اولویت‌دهی موارد آزمون الزاماً مکاشفه ای هستند و از طرفی هیچ‌یک جواب بهینه محسوب نمی‌شوند[۲].

در اغلب فنون اولویت‌دهی موجود، دید محدود شده‌ای وجود دارد و مدلی که محققان در مطالعات تجربیشان برای آزمون رگرسیون استفاده کرده‌اند، آن را به صورت یک آزمون "یک دفعه ای"در نظر گرفته‌اند درحالی‌که مدل مناسب برای آزمون رگرسیون که نشانگر خصوصیات آن باشد، آزمونی "مداوم[۲]" و "دراز مدت" است و که به‌طور سلسله‌وار بعد از هر تغییر در نرم‌افزار اجرا می شود، به‌گونه‌ای که کارایی هر کدام از آن‌ها بر آزمون‌های بعدی مؤثر است.

در همین راستا در[۴] نخستین بار بحث "حافظه دار 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 مجموعه ای از مجموعه آزمون اولیه (بدون کاهش دائمی‌آن) را برای اولویت‌دهی 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 فن پیشینه محور جاری در این است که در اطلاعات پیشینه اجرای موارد آزمون، تنها از "خیر" اجرا شدن یا نشدن مورد آزمون و یا آشکارسازی خطا یا عدم آن در اجرای قبل، آن هم "*به صورت صفر و یک*" در رابطه بازگشتی احتمال انتخاب موارد آزمون در نظر گرفته شده است. در حالی‌که برای اولویت‌دهی کارایی پیشینه محور، 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 مجموعه محک زیمنس پرداخته می‌شود. در

[**] software regression testing
[**] test case prioritization
[**] history-based
[**] historical fault detection performance

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 به صورت نموداری نشان داده شده است. بخش ۵ نیز به نتیجه‌گیری و معرفی کارهای آینده اختصاص دارد.

۲

نخستین بار در سال ۱۹۹۷ مسئله اولویت‌دهی موارد آزمون مطرح گردید[۵] و نخستین تعریف رسمی این مسئله [۳] در سال ۲۰۰۱ توسط رودرمل و گروهش ارائه گردید. این تحقیقات در سال‌های بعد، توسط رودرمل، اِلباوم و چند تن دیگر از محققان این حوزه ادامه یافت[۶،۳-۱۱]. بیشتر فنون اولویت‌دهی موجود، کد محور و بر پایه روش‌های حریصانه‌اند[۸]. انواع دیگر، فنون اولویت‌دهی مدل محور[۱۲] و اولویت‌دهی پیشینه محور[۱۳،۴] 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 بصورت روابط زیر قابل تعریف است:

$$P_0 = h_1$$
$$P_k = \alpha h_k + (1-\alpha) P_{k-1} \qquad k \geq 1,\ 0 \leq \alpha < 1 \qquad (1)$$

در روابط فوق α یک ثابت هموار کننده است و برای وزن‌دهی 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 اجرای آزمون[۶]. برای هر جلسه آزمون i[۵] که مورد آزمون tc در آن اجرا شود، h_i در احتمال انتخاب بعدی مقدار ۰ و در غیر این صورت ۱ 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 مانند بقیه موارد آزمون تضعیف می‌شود و کاراتر بودن موارد آزمون هیچ تأثیری در بالا رفتن اولویت آن‌ها و تسریع انتخابهای بعدیشان ندارد.

۲) کارایی کشف خطای نشان داده شده[۶]. برای هر جلسه آزمون i،که مورد آزمون tc در آن خطا آشکار کند، h_i مقدار ۰ و در غیر این صورت ۱ 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 که بندرت و یا اصلاً خطایی آشکار نکرده‌اند محدود می‌کند.

۳) پوشش مؤلفه‌های برنامه[۷]. منظور از مؤلفه‌های برنامه، جملات، انشعاب‌ها[۸]، مسیرها، توابع، زوجهای تعریف-استفاده و مانند آن است. در این تعریف از H_{tc}، به عنوان مثال اولویت بالاتر به 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 "انتخاب پیشینه محور" در آزمون رگرسیون نرم‌افزاری است و لازمه اولویت‌دهی موارد آزمون این است که از یکی از فنون موجود اولویت‌دهی بر روی نتایج این انتخاب استفاده شود (همانند[۱۳] که با ترکیب دو روش انتخاب مبتنی بر پیشینه و فن اولویت‌دهی هزینه-آگاه[۹] [۱۰]، فنی برای اولویت‌دهی پیشینه-محورِ آگاه از هزینه ارائه داده است).

مشکل روش پیشینه محور موجود این است که انتخاب پارامتر h_k (مجموعه مشاهدات تاریخی اجراهای هر مورد آزمون از اولین تا k اُمین اجرا) که تنها با دو مقدار ۰ یا ۱، آن‌هم فقط بر اساس اجرای مرحله قبل مورد آزمون، در تعیین احتمال انتخاب جاری هر مورد آزمون شرکت می‌کند، ملاک مناسبی نیست. به‌علاوه این‌که مورد آزمونی اخیراً اجرا نشده باشد و یا این‌که در اجرای قبلی خطا آشکار کرده است یا نه، ملاک مناسبی برای بالا رفتن احتمال انتخاب آن در گام اجرای بعدی نیست.

۳

در افزایش اولویت یک مورد آزمون در انتخابهای بعدی چند عامل مؤثر است که یکی از آن‌ها کارایی کشف خطای مورد آزمون در تعداد دفعاتی است که اجرا شده است. مثلاً اگر مورد آزمون tc_A، ۳ خطا در ۲۰ بار اجرا و مورد آزمون tc_B، ۹ خطا در ۱۸ بار اجرا آشکار کرده باشند، کارایی تاریخی نشان می‌دهد مورد آزمون tc_B در کشف خطا بیش از ٪۳۵ نسبت به tc_A بهتر عمل کرده است و اولویت بالاتری بایستی به آن داده شود. مثال فوق نشان می‌دهد که تعداد اجرای موارد آزمون و تعداد دفعاتی که سبب آشکار شدن وجود خطا در نرم‌افزار شده‌اند، توأم با هم و به صورت یک عامل تأثیرگذار در پیشینه کارایی موارد آزمون نقش دارند. بنابراین اگر در k اُمین اجرای آزمون رگرسیون، تعداد دفعاتی که اجرای مورد آزمون tc بر روی نسخه جدید نرم‌افزار با شکست مواجه 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 اجرای آن تا این مرحله را نیز با ec_k نشان دهیم (در محیط محدودیت‌دار تمام موارد آزمون در هر مرتبه اجرا نمی‌شوند)، برای هر مورد آزمون، می‌توان رابطه اولویت مورد آزمون با کارایی آن در k اُمین اجرا را به‌صورت رابطه (۲) نوشت:

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

$$fc_k = \sum_{i=1}^{k-1} f_i \qquad (۳) \qquad ec_k = \sum_{i=1}^{k-1} e_i \qquad (۴)$$

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

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

به عبارت دیگر اولویت پیشینه محور هر مورد آزمون در هر گام، با نسبت آشکارسازی خطا بر کل دفعات اجرای آن متناسب است.

عامل مؤثر دیگر بر اولویت‌دهی موارد آزمون، "سالمندی[۱۰]" آن‌ها است، که به این معنی که موارد آزمون نباید برای مدت طولانی اجرا نشده باقی بمانند. به عبارت دیگر در جریان تکرارهای آزمون رگرسیون باید اطمینان داشته باشیم که همه موارد آزمون بالاخره اجرا شده و خطاهای مربوطه آشکار خواهند شد. در همین راستا عامل h_k در k اُمین اجرا برای هر مورد آزمون به صورت رابطه (۵) تعریف می‌شود:

$$h_k = \begin{cases} 0 & \text{اگر مورد آزمون در جلسه آزمون k-۱ اُم اجرا شده باشد} \\ h_{k-1}+1 & \text{در غیر این صورت} \end{cases} \qquad (۵)$$
$$h_0 = 0$$

در حقیقت عامل h_k مانند یک "شمارنده" عمل می‌کند. در سیستم عامل و مفاهیم مربوط به زمانبندی پردازه مشکلی به‌نام "قحطی‌زدگی[۱۱] پردازش" وجود دارد. اگر در جریان اجرای پردازش‌های متعددِ موجود در سیستم، نوبت اجرا برای مدت طولانی به پردازشی تعلق نگیرد، تعداد این دفعات به‌عنوان "سن" آن در نظر گرفته می‌شود و برای جلوگیری از سالمندی[۱۲] پردازش‌ها، شمارنده‌ای برای سن هر کدام تخصیص می‌یابد. شمارنده h_k نیز نظیر همین نقش را برای دخالت دادن سالمندی هر مورد آزمون در اولویت‌دهی آن دارد. لذا می‌توان 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 جزء موارد آزمون اجرایی قرار بگیرد و اجرا شود. در صورت اجرای مورد آزمون، شمارنده (سن این مورد آزمون) برابر ۰ 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 بصورت 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، بایستی 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 قسمتهای خطادار و فعال کردن و آشکارسازی خطاها در آن‌ها بیشتر است، لذا اولویت بالاتری به آن 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 آنچه در روابط (۵)، (۶) و (۷) گفته شد، می‌توان رابطه اولویت‌دهی در k اُمین گام اجرا برای هر مورد آزمون را به صورت رابطه (۸) نوشت:

$$PR_k = \alpha \frac{fc_k}{ec_k} + \beta\, PR_{k-1} + \gamma\, h_k \qquad (۸)$$
$$0 \le \alpha, \beta, \gamma < 1,\ k \ge 1$$

با تغییر ثوابت هموارکننده α، β و γ در رابطه فوق، می‌توان به کنترل میزان تأثیر عوامل مؤثر در اولویت‌دهی پرداخت. باید توجه کرد که ضریب h_k (γ) باید بسیار کوچکتر از سایر ضرایب α و β در نظر گرفته شود، زیرا h_k برحسب شرایط اجرای مورد آزمون هر مرتبه یک واحد افزایش می‌یابد و باید به طریقی اثر آن را در مقابل ec_k / fc_k و PR_{k-1} که هر دو مقادیری کسری هستند، کنترل نمود که به اشتباه تأثیر 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.

برای آشکارتر شدن بیشتر عملکرد روش پیشنهادی یک برنامه ساده و دنباله آزمون آن را بعنوان مثال مورد بررسی قرار می‌دهیم (شکل ۱). برنامه فوق پنج عدد صحیح را به‌عنوان ورودی می‌گیرد و محاسبات آن به‌گونه‌ای است که در انشعاب‌های مختلف احتمال خطای تقسیم برصفر وجود دارد. برای این برنامه که با تغییرات جزئی از [۱۴] اقتباس شده است، یک مجموعه آزمون با پوشش کافی برای

```
۱: read (a, b, c, d)                B۳: if (c>۰)
B۱: if (a>۰)                        B۴:   if (d>۰)
۲:      x=۲;                        B۵:     if (e>۰)
۳:   else                           ۱۲:       output (۱/ (b+۱));
۴:     begin                        ۱۳:     else
۵:      x=۵;                        ۱۴:       output (۱/(y-۴));
B۲:    if (b>۰)                     ۱۵:     endif
۶:       y=x+۱;                     ۱۶:   else
۷:     else                         ۱۷:     output (۱/(x-۵));
۸:       y=x-۱;                     ۱۸:   endif
۹:     endif                        ۱۹: else
۱۰:  end                            ۲۰:   output (۱/(x-۵));
۱۱: endif                           ۲۱: endif
```

شکل۱ : برنامه نمونه

انشعاب[۱۳] در نظر گرفته شده است (تعداد کافی مورد آزمون برای اجرای تمامی شاخه‌های if و else برنامه).

اطلاعات مربوط به موارد آزمون و پوشش آن‌ها در جدول ۱ آمده است. ستون دوم این جدول، ورودی‌های برنامه را مشخص می کند که در واقع موارد آزمون برنامه شکل ۱ هستند. این دنباله متشکل از هشت مورد آزمون است که برای پوشش (اجرای) تمامی انشعاب‌های برنامه کافی است. هر کدام از B_i^T و B_i^F ها در ستون‌های جدول، درست یا نادرست ارزیابی شدن انشعاب i را نشان می دهند و سطرهای علامت‌گذاری شده نیز پوشش مورد آزمون را نسبت به درستی یا نادرستی ارزیابی انشعاب خاص نشان می دهد.

باید توجه کرد که پوشش انشعاب قویتر از پوشش جملات است و در پوشش انشعاب، باید هر دو حالت T و F برای تمام انشعاب‌های برنامه آزموده شود.

در جدول۲ حاصل به‌کارگیری رابطه اولویت‌دهی روی برنامه شکل ۱ در چهارگام اجرای آزمون خلاصه شده است. در این جدول هشت ستون سمت راست، مقادیر محاسبه شده توسط رابطه (۸) برای هر مورد آزمون در آزمون‌های رگرسیون را نشان می‌دهد. ستون‌های سمت چپ نیز به ترتیب دفعات آزمون رگرسیون و اولویت موارد آزمون را نشان می‌دهند. این اولویت‌ها با مرتب کردن مقادیر متناظر با هر مورد آزمون در هر سطر بدست می‌آیند. سپس برنامه شکل ۱ با استفاده از موارد آزمون اولویت‌دهی شده اجرا می‌شود. بنابر شرایط، ممکن است با اجرای موارد آزمون خطایی آشکار شود یا نشود. در خاتمه هر مرحله اجرای آزمون، خطاهای آشکارشده رفع شده و مقادیر جدید اولویت برای موارد آزمون توسط رابطه (۸) محاسبه می‌شوند. همان‌طور که در جدول ۲ ملاحظه می‌شود، مقادیر اولویت موارد آزمون در آزمون اول، درصد پوشش انشعاب هر مورد آزمون است. به این ترتیب موارد آزمون ۴، ۵ و ۸ دارای بالاترین اولویت و موارد آزمون۱ و ۶ دارای پایین‌ترین اولویت هستند. به دلیل اینکه در آزمون رگرسیون زمان کافی برای اجرای همه موارد آزمون وجود ندارد و اولویت‌دهی نیز به همین دلیل انجام می‌شود، در این مثال فرض می‌کنیم که هر دفعه ۵۰ درصد از موارد آزمون اجرا شوند. پس در اولین آزمون، موارد آزمون ۴، ۵، ۸ و ۲ اجرا می‌شوند و منجر به آشکارشدن خطاهای خطوط ۱۲، ۱۴ و ۱۷ می‌گردند. باید توجه داشت که مورد آزمون ۴ اگرچه اجرا شده است امّا خطایی را آشکار نساخته است. ازاین‌رو در جلسه آزمون بعد، fc_1 برای مورد آزمون ۴، صفر و برای موارد آزمون ۵، ۸ و ۲، یک خواهد بود. ضمن اینکه ec_1 برای این چهار مورد آزمون یک می‌باشد. برای سایر موارد آزمون که اجرا نشده‌اند، fc_1 و ec_1 هر دو صفر می‌گردد. همچنین مقدار h_1 برای موارد آزمونی که اجرا شده‌اند صفر و برای بقیه یک می‌شود. برای آزمون بعدی برنامه، مقادیر جدید اولویت محاسبه شده و موارد آزمون اولویت‌دهی و روی برنامه اجرا می‌گردند. دراین مثال چهار مرتبه آزمون اجرا و موارد آزمون اولویت‌دهی شده‌اند. با دقت به اولویت‌های آزمون دوم، مشاهده می‌شود که چون موارد آزمون ۶، ۱، ۷ و ۳ در آزمون اول اجرا نشده‌اند، اولویت آن‌ها بیشتر شده است. در این آزمون، موارد آزمون ۵، ۸، ۲ و ۳ اجرا می‌شوند، اما فقط مورد آزمون ۳ خطای خط ۲۰ را کشف می‌کند. به همین سبب در آزمون سوم، مورد آزمون ۳ بالاترین اولویت را کسب کرده است. به‌علاوه موارد آزمون ۷، ۱ و ۶ چون در آزمون قبل اجرا نشده‌اند، اولویت بالاتری پیدا کرده‌اند. موارد آزمون ۵، ۸ و۲ نیز در مرحله دوم اجرا شده ولی چون خطایی کشف نکرده‌اند، اولویت آن‌ها کاهش یافته است. در آزمون چهارم نیز اولویت موارد آزمونی که در آزمون سوم اجرا نشده‌اند، افزایش یافته و اولویت موارد آزمونی که در آزمون سوم اجرا شده‌اند، کاهش یافته است. به این ترتیب رابطه (۸) با استفاده از سه عامل، اولویت هر مورد آزمون را محاسبه می‌نماید: (۱) نسبت خطاهای آشکار شده توسط هر مورد آزمون به دفعات اجرای آن از ابتدای آزمون رگرسیون، (۲) مقدار اولویت در آزمون قبلی و (۳) دفعاتی که هر مورد آزمون به‌علت اولویت کم اجرا نشده است.

٤- ارزیابي روش پیشنهادي

٤-١- متریک ارزیابي

در سال ۱۹۹۷ نخستین تعریف رسمی از مسئله اولویت‌دهی موارد آزمون و نیز متریکِ APFD که متوسط وزن‌دار درصد خطاهای کشف شده در اجرای مجموعه آزمون[۱۴] است، برای ارزیابی فنون اولویت‌دهی آزمون از لحاظ احتمال کشف زودتر خطاها ارائه شد[۲].

فرمول محاسبه APFD در اصل به صورت زیر است:

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

در رابطه (۹)، n تعداد موارد آزمون و m تعداد خطاهای موجـود را نشان می دهد. هر TF_i در رابطه فوق محل مـورد آزمـونی را در دنبالـه مرتب موارد آزمون در رشته اولویت‌دهی شده نشان می‌دهـد کـه بـرای اولین بار خطای i اُم را آشکار نموده است.

به بیان ساده می توان گفت که هرچه مقدار محاسبه شده APFD برای فن اولویت‌دهی به صد درصد نزدیکتـر باشـد، کـارایی آن فـن در کشف سریعتر خطاها بیشتر خواهد بود.

برای درک بهتر معمولاً "درصد خطاهای آشکارشده" نسبت به "کسر

جدول۱: پوشش انشعاب موارد آزمون

جمله خطادار	خط خروج	پوشش انشعاب	B_5^F	B_5^T	B_4^F	B_4^T	B_3^F	B_3^T	B_2^F	B_2^T	B_1^F	B_1^T	ورودی (a,b,c,d,e)	مورد آزمون
-	۲۰	۲۰٪					×					×	(۱.۱,-۱.۰.۰)	۱
۱۷	۱۷	۴۰٪			×				×	×	×		(-۱,-۱.۱,-۱.۰)	۲
۲۰	۲۰	۳۰٪					×			×	×		(-۱.۱,-۱.۰,-۱)	۳
-	۱۲	۵۰٪		×		×		×		×	×		(-۱.۱.۱.۱.۱)	۴
۱۲	۱۲	۵۰٪		×		×		×	×		×		(-۱,-۱.۱.۱.۱)	۵
-	۲۰	۲۰٪					×					×	(۱, -۱,-۱,-۱,-۱)	۶
۲۰	۲۰	۳۰٪					×			×	×		(-۱.۱,-۱.۱.۰)	۷
۱۴	۱۴	۵۰٪	×			×		×	×		×		(۰.۰.۱.۱,-۱)	۸

جدول۲: مقادیر محاسبه شده برای اولویت موارد آزمون در هریک از آزمون‌های رگرسیون

مورد آزمون ۱	مورد آزمون ۲	مورد آزمون ۳	مورد آزمون ۴	مورد آزمون ۵	مورد آزمون ۶	مورد آزمون ۷	مورد آزمون ۸	اولویت موارد آزمون از چپ به راست	دفعات آزمون رگرسیون
۰.۲	۰.۴	۰.۳	۰.۵	۰.۵	۰.۲	۰.۳	۰.۵	[۴,۵,۸,۲,۳,۷,۱,۶]	۱
۰.۳۵	۰.۷	۰.۴	۰.۲۵	۰.۷۵	۰.۳۵	۰.۴	۰.۷۵	[۵,۸,۲,۳,۷,۱,۶,۴]	۲
۰.۶۷۵	۰.۶	۰.۷	۰.۳۷۵	۰.۶۲۵	۰.۶۷۵	۰.۷	۰.۶۲۵	[۳,۷,۱,۶,۵,۸,۲,۴]	۳
۰.۳۳۷۵	۰.۸	۰.۶	۰.۶۸۷۵	۰.۸۱۲۵	۰.۳۳۷۵	۰.۳۵	۰.۸۱۲۵	[۵,۸,۲,۴,۳,۷,۱۶]	۴

اجرای شده از آزمون" در یک نمودار باهم نمایش داده می‌شود.

در این نمودار "مساحت زیر نمودار" نشان‌دهنده مقدار "متوسط درصد خطاهای آشکار شده در یک دور اجرای آزمون رگرسیون" است. هر چه رشته اولویت‌دهی شده خطاها را سریعتر آشکار کند، این سطح و مقدار APFD متناظر با آن بزرگتر است.

٤-۲- مطالعات تجربي

استفاده از مجموعه زیمنس[۱۵][۱۵] که شامل هفت برنامه به زبان C است، در "آزمایش‌های کنترل شده" 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 مورد آزمون برابر با زیر مجموعه‌های 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 نسخه‌های چندخطایی نیاز است. به همین دلیل، مجموعه‌ای ترکیبی از خطاهای دو بدو مجزا ایجاد شده است. این خطاها بنابرتجارب برنامه‌نویسان از انواع خطاهای متداول ایجاد شده و به خطاهای واقعی بسیار نزدیک هستند.

مطالعه تجربی دیگر، بررسی موردی است که بر روی برنامه محک space صورت گرفته است، که برنامه‌ای بزرگ(۱۰KLOC) 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 آزمایش‌های کنترل شده و بررسی موردی انجام شده، از مجموعه‌های چند خطایی، 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. همچنین ضرایب α و β نیز مقداری تصادفی بین ۰.۷ و ۰.۹ و ضریب γ مقداری تصادفی بین ۰.۱ و ۰.۰۴ انتخاب 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 کمتر موارد آزمون در واقع به "صرفه‌جویی در منابع آزمون" نیز کمک کرده است.

برای مقایسه و نمایش کارایی روش اولویت‌دهی پیشنهادی، از روش متداول در مطالعات تجربی، یعنی "نمودار جعبه‌ای[۱۶]" استفاده شده است[۸،۷،۶،۳،۲] 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 جعبه همرنگ، نتایج متوسط کشف خطا (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[۱۶] 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 نیز حداقل و حداکثر داده‌هایی را که پرت و برون‌هشته[۱۷] نیستند نشان می‌دهد.

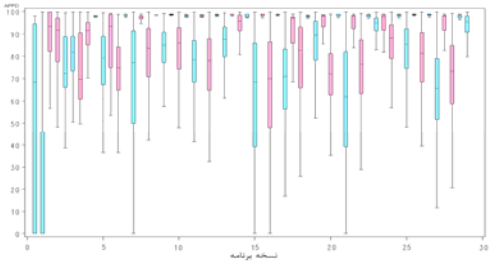
در نمودار جعبه‌ای، گستره جعبه‌ها در امتداد محور عمودی(APFD)، گستره مقادیر متوسط کشف خطا را نشان می‌دهد. هر چه گستردگی جعبه کمتر باشد، به این معناست که عملکرد روش اولویت‌دهی، پایداری بیشتری دارد و رفتار یکنواخت‌تری از نظر سرعت کشف خطا از خود نشان می‌دهد. روشن است که بالاتر بودن محل جعبه (بیشتر بودن مقادیر APFD)، حاکی از قدرت روش اولویت‌دهی در کشف سریعتر خطا است. 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 جعبه)، بهتر عمل کرده است.

5- نتیجه گیري و کارآینده

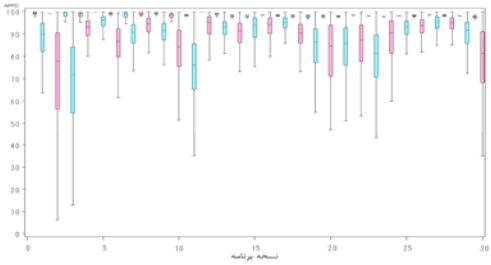
در این مقاله روش جدیدی برای اولویت‌دهی مبتنی برپیشینه موارد آزمون ارائه شد که با در نظرگرفتن محدودیت‌های واقعی محیط اجرا و هزینه زیاد اجرای تکراری کل موارد آزمون، تنها بخشی از دنباله آزمون اولویت‌دهی شده را اجرا می‌کند.

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.

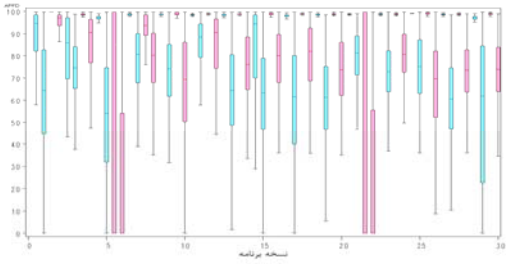
در ادامه این کار با توجه به این که از مجموعه زیمنس برای ارزیابی استفاده شده است و آزمایش‌ها کنترل شده هستند، می‌توان مطالعات تجربی بیشتری با برنامه‌های واقعی ترتیب داد تا اثر روش پیشنهادی در محیط‌های واقعی آشکارتر گردد. به‌علاوه می‌توان با استفاده از اطلاعات اجرایی در هر مرحله ضرایب رابطه (۸) را دقیق‌تر محاسبه کرد تا بهترین اولویت بدست آید.



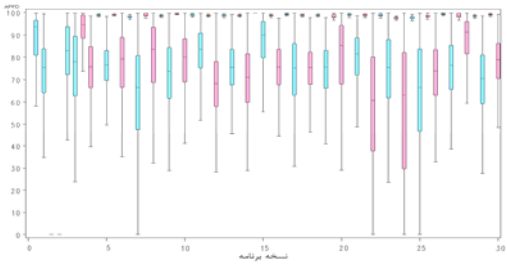
الف) مطالعه تجربی: برنامه print-tokens



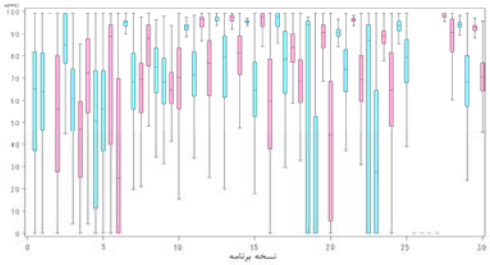
ب) مطالعه تجربی: برنامه print-tokens۲



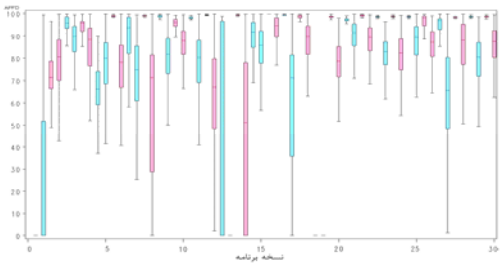
ج) مطالعه تجربی: برنامه schedule



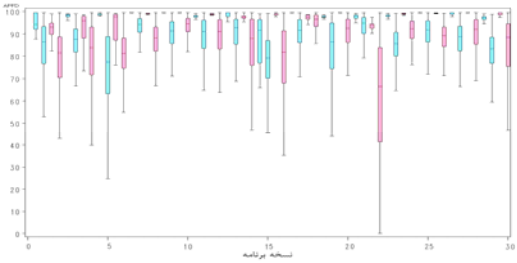
د) مطالعه تجربی: برنامه schedule۲



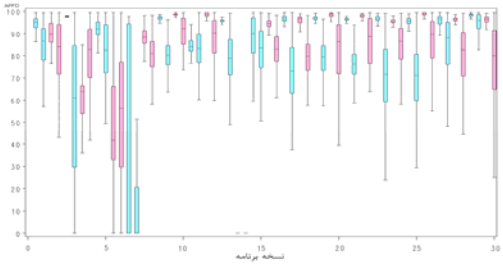
ه) مطالعه تجربی: برنامه tcas



و) مطالعه تجربی: برنامه **replace**



ز) مطالعه تجربی: برنامه **tot-info**



ح) بررسی موردی: برنامه**space**

شکل۲) نمودار جعبه‌ای مقایسه روش پیشنهادی با اولویت تصادفی.

مراجع

[۱] Burnstein, I. *Practical software testing : a process-oriented approach.* ISBN ۰-۳۸۷-۹۵۱۳۱-۸, Springer-Verlag New York, Inc, ۲۰۰۳.

[۲] Rothermel, G., Untch, R. H., Chu, C., Harrold, M. J., *"Test case prioritization: an empirical study"*, Proc. IEEE Int. Conf. on Software Maintenanc. Oxford, England, pp. ۱۷۹-۱۸۸, ۱۹۹۹.

[۳] Rothermel, G. R., Untch, H., Chu, C. Harrold, M. J., *"Prioritizing Test Cases for Regression Testing"*, Proc. IEEE Transactions on Software Engineering. pp. ۱۰۲-۱۱۲, ۲۰۰۱.

[٤] Kim, J. M., Porter, A. *"A History-Based Test Prioritization Technique for Regression Testing in Resource Constrained Environment"*, *Proc. ۲۴th Int'l Conf. Software Engineering*. pp. ۱۱۹-۱۲۹, ۲۰۰۲.

[۵] Wong, W. E., Horgan, J. R., London, S., Agrawal, H., *"A study of effective regression testing in practice"*, Proc. ۸th Int'l Symp. Software Reliability Engineering. pp. ۲۳۰-۲۳۸, ۱۹۹۷.

[٦] Elbaum S., Malishevsky, A. G., Rothermel, G., *"Test Case Prioritization: A Family of Empirical Studies"*, Proc. IEEE Transactions on Software Engineering, Vol. ۲۸, No. ۲, pp. ۱۵۹-۱۸۲, ۲۰۰۲.

[۷] Elbaum, S., Malishevsky, A., Rothermel, G., *"Prioritizing test cases for regression testing"*, Proc. ۷th Int'l Syrup. Software Testing and Analysis. pp. ۱۰۲-۱۱۲, Aug. ۲۰۰۰.

[۸] Li, Z., Harman, M., Hierons, R., *"Search Algorithms for Regression Test Case Prioritization"*, Proc. IEEE Transactions on Software Engineering, Vol. ۳۳, pp. ۲۲۵-۲۳۷, ۲۰۰۷.

[۹] Malishevsky, A. G., Elbaum, S., Rothermel, G., Kanduri, S., *"Selecting a Cost-Effective Test Case Prioritization Technique"*, Software Quality Control, Vol. ۱۲, pp. ۱۸۵-۲۱۰, ۲۰۰۴.

[۱۰] Malishevsky, A. G., Ruthruff, J. R., Rothermel, G., Elbaum S., *"Cost-cognizant test case prioritization"*, Department of Computer Science and Engineering, Nebraska, Lincoln, Tech. Rep. TR-UNL-CSE-۲۰۰۶-۰۰۰۴ March ۲۰۰۶.

[۱۱] Srivastava, P. R., *"Test Case prioritization"*, JATIT, Computer Science and Information System Group, BITS Pilani, India-۳۳۳۰۳۱, ۲۰۰۵ – ۲۰۰۸.

[۱۲] Korel, B., Koutsogiannakis, G., Tahat, L. H., *"Model-based test prioritization heuristic methods and their evaluation"*, In Proc. ۳rd Int'l Workshop on Advances in Model-Based Testing, London, UK, July ۲۰۰۷.

[۱۳] Park, H., Ryu, H., Baik, J., *"Historical Value-Based Approach for Cost-Cognizant Test Case Prioritization to Improve the Effectiveness of Regression Testing"*, paper appears in: ۲th Int'l Conf. Secure System Integration and Reliability Improvement, pp. ۳۹-۴۶, Japan, ۲۰۰۸.

[۱٤] Jeffrey, D., Gupta, N., *Improving Fault Detection Capability by Selectively Retaining Test Cases during Test Suite Reduction.* Source IEEE Transactions on Software Engineering archive, Pages ۱۰۸-۱۲۳, ۲۰۰۷.

[۱۵] Rothermel, G., Elbaum, S., Kinneer, A., Do, H., *Software-artifact infrastructure repository*, http://www.cse.unl.edu/~galileo/sir.

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

---

[۱] fitness metric

[۲] continuous

[۳] history-based regression test selection

[٤] test execution history

[۵] test session i

[٦] demonstrated fault detection effectiveness

[۷] coverage of program entities

[۸] branches

[۹] cost-cognizant test case prioritization

[۱۰] ageing

[۱۱] starvation

[۱۲] ageing

[۱۳] branch coverage adequate

[۱٤] Average Percentage of Fault Detection

[۱۵] Siemens suite programs

[۱۶] boxplot
[۱۷] outlier