دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده مهندسی برق و کامپیوتر

موضوع پروژه

تعریف تابع سازواری مناسب برای آزمون جستجو محور مبتنی بر توصیف تابعی

پایان‌نامه برای دریافت درجه کارشناسی

در رشته مهندسی کامپیوتر تمرکز نرم‌افزار

نام دانشجو

سیده مهرناز شمس‌آبادی

شماره دانشجویی

۸۱۰۱۹۶۴۹۳

استاد راهنما:

دکتر رامتین خسروی

بهمن‌ماه ۱۴۰۰


بسم الله الرحمن الرحيم

تعهدنامه اصالت اثر

باسمه تعالی

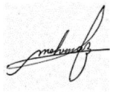
اینجانب سیده مهرناز شمس‌آبادی **تایید می کنم که مطالب مندرج در این پایان نامه حاصل تلاش اینجانب است و به دستاوردهای پژوهشی دیگران که در این نوشته از آنها استفاده شده است مطابق مقررات ارجاع گردیده است. این پایان نامه قبلاً برای احراز هیچ مدرک هم سطح یا بالاتر ارائه نشده است.**

کلیه حقوق مادی و معنوی این اثر متعلق به دانشکده فنی دانشگاه تهران می باشد.

نام و نام خانوادگی دانشجو :

سیده مهرناز شمس‌آبادی

امضای دانشجو :



با تشکر از دکتر رامتین خسروی، دکتر احسان خامس‌پناه، مهندس آروین ذاکریان و مهندس هادی صفری که بنده را در راستای انجام این پژوهش یاری و راهنمایی کردند.

چکیده

تولید خودکار آزمون[1] برای تایید صحت منطق نرم‌افزارهایی که پیچیدگی و قابلیت زیادی دارند، به دلیل تعدد نیازمندی‌های غیرکارکردی، با دشواری همراه است. استفاده از روش‌های آزمون مبتنی بر توصیف‌های تابعی[2] می‌توانیم پس از مدل کردن منطق کارکرد سیستم تحت آزمون با استفاده از زبان‌های برنامه‌نویسی تابعی، به کمک روش آزمون جستجو محور[3]، آزمایه‌هایی به صورت خودکار تولید کنیم. تعریف تابع سازواری[4] مناسب نقش کلیدی در کیفیت آزمایه‌های تولیدی دارد. در این پروژه سامانه هسته معاملات بورس به عنوان سیستم مبنا در نظر گرفته شده‌است. طی پژوهش‌هایی که پیش از این روی این سامانه انجام شده‌است، منطق این سامانه با استفاده از زبان برنامه‌نویسی هسکل[5] مدل شده و با استفاده از این مدل، الگوریتمی جستجو محور برای تولید آزمایه‌های خودکار پیاده‌سازی شده‌است. با توجه به اهمیت تابع سازواری، هدف در این پروژه، بهبود این تابع بوده‌است. تابع سازواری طی پژوهش‌های قبلی بر مبنای میزان پوشش توصیف مبدا تعریف شده‌است. در این پروژه با بررسی میزان تاثیر اندازه‌ی مجموعه آزمایه بر روی عملکرد تابع سازواری و یافتن وزنی بهینه برای افزودن آن به فرمول سازواری، توانستیم عملکرد آن را بهبود ببخشیم.

کلمات کلیدی:

تابع سازواری، اندازه مجموعه آزمایه، میزان پوشش، تولید خودکار آزمایه

[1] Automatic test case generation

[2] Specification-based testing

[3] Search-based

[4] Fitness Function

[5] Haskell

فهرست مطالب

فهرست شکل‌ها

فهرست جدول‌ها

فهرست علائم اختصاری

GA	Genetic Algorithm
ME	Matching Engine
SUT	System Under Test

فصل ۱: مقدمه و بیان مسئله

در این فصل ابتدا مقدمه‌ای بر تاریخچه‌ی پژوهش آمده و سپس مسئله شرح داده شده‌است و در نهایت نیز به بیان روش کلی انجام پروژه پرداخته می‌شود.

۱-۱- مقدمه

این پروژه در ادامه‌ی یک پژوهش است که پروژه‌ی مبنا در این پژوهش، سامانه هسته معاملات بورس است که کارکرد اصلی آن، بخش ME در بازار سرمایه است. این سامانه نیازمندی‌های غیرکارکردی و کیفیتی قابل توجهی مثل کوتاه بودن زمان پاسخ‌دهی به درخواست‌ها یا دسترس‌پذیری بالا دارد. مجموعه‌ی این نیازمندی‌ها سبب ورود عناصر غیردامنه‌ای و غیرکارکردی به سیستم می شود حتی ممکن است معماری سیستم را دچار تغییرات اساسی کند. با این که افزودن این عناصر سبب بهتر شدن کیفیت سیستم می‌شود اما ترکیب آن‌ها با هم ممکن است به درستی کارکرد سیستم صدمه بزند. از طرفی برای این که این سیستم در بازار سرمایه مورد استفاده قرار گیرد، لازم است از درستی عملکرد آن اطمینان کافی داشته باشیم. برای اطمینان از عملکرد این سامانه که به زبان جاوا توسعه داده‌شده‌است، نیازمند آزمون کافی هستیم اما به سبب پیچیدگی دامنه، اتکا به آزمایه‌های یکه[1] و یکپارچه‌ی[2] توسعه داده‌شده توسط آزمون‌گرها اطمینان کافی را در اختیار ما قرار نمی‌دهند. بنابراین آزمون‌گرها به سمت استفاده از روش‌های توصیف تابعی برای آزمون سیستم رفتند. [1] بنابراین ابتدا باید منطق SUT با استفاده از یک زبان برنامه‌نویسی تابعی مدل شود. با توجه به این که هسکل یه زبان برنامه‌نویسی تابعی محض، همه‌منظوره و statically typed[3] است، برای مدل کردن منطق سامانه هسته معاملات بورس از آن استفاده شده‌است. سپس با استفاده از یک روش جستجو محور برای این مدل هسکل به صورت آزمایه تولید شده‌است. برای این منظور یک اسکریپت[4] پایتون[5] نوشته شده‌است که با استفاده از یک GA، مجموعه آزمایه‌ای تولید می‌کند. تابع سازواری تعریف شده برای GA از نتایج hpc[6] و پوشش بولی[7] هسکل استفاده کرده‌است. بنابراین الگوریتم سعی می‌کند تا مجموعه آزمایه‌ای را به عنوان خروجی بدهد که بیشینه میزان پوشش روی کد هسکل را داراست. در نهایت

[1] Unit test

[2] Integration test

[3] http://learnyouahaskell.com/introduction

[4] script

[5] Python

[6] ابزاری برای اندازه‌گیری میزان پوشش روی یک برنامه هسکل است.

[7] Boolean Coverage

باید ورودی‌ها و خروجی‌های به دست آمده را به پروتکل ورودی و خروجی SUT تبدیل کنیم تا بتوانیم این آزمایه‌ها را روی آن اجرا کنیم. به این منظور نیز مترجم‌های مورد نظر نیز توسعه داده‌شده‌است. همچنین اسکریپت‌هایی برای مقایسه‌ی خروجی‌های مدل و خروجی‌های SUT برای بررسی صحت کارد سیستم تحت آزمون نوشته شده‌است. نتایج این پژوهش بسیار قابل توجه بوده‌است. آزمایه‌های تولید شده، منجر به تشخیص باگ‌های بحرانی در سیستم تحت آزمون شده‌است. بنابراین این پژوهش ادامه یافت تا فرآیند تولید آزمایه‌ها بهبود یابد. در فصل ۲ توضیحات بیشتری پیرامون بخش‌های مختلف این پژوهش ارائه می‌شود. در این پروژه، هدف بهبود تابع سازواری با بررسی نقش اندازه مجموعه آزمایه در آن است.

۲-۱- بیان مسئله

مسئله‌ی پروژه‌ی حاضر، بهبود تابع سازواری است. مهم‌ترین بخش آن نیز بهبود عملکرد آن با بررسی بیشتر تاثیر اندازه‌ی مجموعه آزمایه است. البته تابع سازواری را از جهات مختلف می‌توان بهبود بخشید؛ زیرا اگرچه هدف نهایی رسیدن به میزان پوشش بیشتر بر روی SUT است، زمان اجرای GA نیز از اهمیت برخوردار است. همچنین در صورت کاهش تعداد آزمایه‌های یک مجموعه آزمایه (یا همان اندازه مجموعه آزمایه)، زمان آزمون این مجموعه بر روی SUT با کاهش همراه خواهد بود. کاهش زمان آزمون سیستم مفید است اما در مورد سیستمی که در این پژوهش به آن اختصاص دارد، به سبب کوتاهی زمان پاسخ‌دهی سیستم به درخواست‌ها و کارایی بالای آن، زمان آزمون برای مجموعه آزمایه‌ای با حداکثر ۴۰ آزمایه (و هر آزمایه شامل حداکثر ۱۰ درخواست است.)، همواره ناچیز است.

با توجه به نتایجی که در فصل ۲ خواهید دید، به منظور بهبود تابع سازواری، می‌توان میزان تاثیر اندازه مجموعه آزمایه را در سازواری بررسی کرد تا علاوه بر رسیدن به یک ضریب بهینه برای آن در محاسبه‌ی مقدار سازواری، بتوانیم زمان اجرای GA را کاهش دهیم. علاوه بر این به منظور افزایش میزان پوشش در SUT می‌توان ترکیب انواع معیارهای پوشش در سازواری را بررسی کرد؛ زیرا با اتکا به نتایج می‌توان گفت که اگرچه استفاده از هر معیار به عنوان سازواری پوشش قابل توجه و بعضا خوبی در SUT می‌دهد، اما تا رسیدن به پوشش قابل قبول فاصله وجود دارد. به این ترتیب بررسی این فرض که ترکیب معیارهای پوشش مختلف، می‌تواند تاثیر مثبتی بر میزان پوشش بر روی SUT داشته باشد یا نه، نیز از اهداف پروژه است.

۳-۱- روش انجام پروژه

با استفاده از تابع سازواری که در پژوهش‌های پیشین پیاده‌سازی شده‌است[1]، می‌توان با تغییر مقدار سازواری، اجرای GA و سپس بررسی میزان پوشش حاصل از مجموعه آزمایه‌ی خروجی بر روی مدل و همچنین SUT سازواری‌های مختلف را با هم مقایسه کرده تا در نهایت به تابع سازواری بهینه‌تری بر سیم. برای به دست آوردن میزان پوشش شاخه‌ای روی SUT نیز از ابزار JaCoCo[2] استفاده می‌کنیم. به منظور محدود کردن عمق فرآیند جستجو، برای اندازه‌ی مجموعه آزمایه، تعداد درخواست‌های داخل یا آزمایه و همچنین پارامترهای مختلف یک درخوا ست، محدودیت‌هایی تعریف شده‌ا ست. محدودیت اندازه‌ی مجموعه آزمایه برابر ۴۰ و محدودیت حداکثر تعداد درخواست‌های یک آزمایه برابر ۱۰ تعریف شده‌است. جمعیت اولیه در GA برابر ۱۰۰ و حداکثر تعداد تکرار آن برابر ۱۰۰۰ است. سایر پارامترهای GA نیز مانند بخش ۳-۳-۲ می‌باشد. نتایج حاصل از این بررسی‌ها را در فصل ۳ بیان شده‌است.

[1] https://github.com/functional-specification-based-testing/search-based-test-suite-generator.git

[2] https://www.jacoco.org/

فصل ۲: تاریخچه پژوهش

در این فصل به شرح پژوهش‌هایی که پیش از این انجام شده و نتایج حاصل از آن پرداخته می‌شود.

۲-۱- سیستم تحت آزمون

SUT در پژوهش، بخش ME از سامانه هسته معاملات بورس است که اکنون در شرکت رادین بورس، در حال توسعه می‌باشد. این سیستم انواع مختلفی از سفارش‌ها در بازار سرمایه از جمله سفارش‌هایی که از نظر مقدار و زمان، دارای شرایط خاص هستند. همچنین قبل و بعد از معامله، موارد و شرایطی مثل میزان دارای خریدار و سهام مجاز برای یک سهام‌دار را بررسی می‌کند. از نظر نیازمندی‌های غیرکارکردی، زمان پاسخ‌دهی به درخواست‌ها نیز در این سیستم کوتاه و در حد میلی‌ثانیه است. توسعه‌دهندگان برای پیاده‌سازی این سیستم با توجه به ویژگی‌های آن، پیچیدگی‌هایی وارد آن کرده‌اند. برای مثال برای داشتن کارایی مناسب از Disruptor framework[1] استفاده کرده‌اند. همچنین این سیستم دارای ساختار داده‌هایی برای مدیریت حجم زیاد داده‌های در حافظه می‌باشد.

۲-۲- توصیفات تابعی

روش آزمون مبتنی بر توصیفات تابعی به آزمون مبتنی بر رفتار یا آزمون جعبه سیاه نیز شناخته می‌شود؛ زیرا در این نوع آزمون، آزمونگران بر آنچه سیستم انجام می‌دهد و نه نحوه‌ی انجام آن تمرکز و توجه می‌کنند. این روش از توصیفات سیستم به عنوان مرجعی برای انتخاب داده‌های آزمون استفاده می‌کند و این توصیفات ممکن است هر چیزی مانند اسناد، موارد استفاده، مجموعه‌ای از مدل‌ها یا یک نمونه‌ی اولیه باشد. بنابراین برای زبان توصیفی که برای تولید آزمایه استفاده می‌شود، چندین نیازمندی وجود دارد. [2] این زبان باید برای بیان نیازمندی‌های دامنه به اندازه‌ی کافی روشن و رسا باشد و همچنین دقیق و بدون ابهام تعریف شود. به منظور تعیین این که SUT چه چیزی را محاسبه می‌کند (و نه اینکه چگونه آن را محاسبه می‌کند.) باید از انتزاع نیز پشتیبانی شود. به این ترتیب برای توصیف یک سیستم به منظور تولید آزمایه، استفاده از یک زبان تابعی محض منا سب ا ست؛ زیرا تمامی نیازهای زبان تو صیفی مورد نظر روش‌های مبتنی بر تو صیف تابعی را برآورده می‌کنند. در ابتدا با توجه به همه‌منظوره بودن زبان‌های تابعی، در صنعت استفاده می‌شوند و بدیهی است که برای بیان نیازمندی‌های دامنه به اندازه‌ی کافی، گویا هستند. همچنین انواع داده‌های جبری، امکان محاسبات بازگشتی روی لیست‌ها و الگوهای انتزاعی غنی روی بر پایه‌ی توابع بالاتر که مشخصات و ویژگی‌های

[1] http://lmax-exchange.github.io/disruptor/disruptor.html

سطح بالای سیستم را فعال می‌کند. در نهایت با توجه به قواعد معنایی رسمی[1] این زبان‌ها، توصیفات دقیق و بدون ابهام خواهند بود. بنابراین برای توصیف SUT در این پژوهش، از زبان تابعی هسکل استفاده شده‌است.[2] علاوه بر استفاده‌ی قابل توجه هسکل در صنعت و آکادمی، این زبان توسط ابزارهای مختلف از جمله ابزارهای اندازه‌گیری میزان پوشش مبدا، پشتیبانی می‌شود.

بخش‌های ۳-۲ و ۴-۲ در ادامه‌ی پژوهش [1] هستند ولی هنوز به اتمام نرسیده‌اند.

۳-۲- تولید آزمایه

در تولید خودکار آزمایه، هدف تولید مجموعه آزمایه‌ای است که بیشترین میزان پوشش را بدهد. به این منظور یک الگوریتم جستجو برای جستجوی فضای ورودی استفاده شده است تا مجموعه تستی بهینه ارائه شود که میزان پوشش بیشتری نیز دارد. برای الگوریتم جستجو نیز از GA که روشی برای تولید آزمایه بر پایه‌ی روشی Metaheuristic[3] است، استفاده شده‌است. [3] قابل ذکر است که میزان پوشش در این فرآیند نه بر پایه‌ی مشخصات ساختاری و رفتاری SUT که بر پایه‌ی توصیف سیستم است. در ادامه در مورد تشریح بیشتر نحوه‌ی تولید آزمایه‌ها می‌پردازیم.

۱-۳-۲- شرح مسئله

ابتدا باید روشن شود که از GA چه خروجی می‌خواهیم. ما در نهایت مجموعه‌ای از آزمایه‌های معتبر می‌خواهیم که آن را به SUT (که مانند یک جعبه سیاه در این پژوهش با آن سر و کار داریم.) و مدلی که آن را توصیف می‌کند (مدلی که در هسکل پیاده‌سازی شده‌است.) بدهیم و در نهایت بتوانیم با مقایسه‌ی خروجی‌های دو سیستم از درستی یا نادرستی سناریوهای مختلف در SUT مطلع شویم. هر آزمایه باید ابتدا دارای پارامترهایی برای مقداردهی اولیه‌ی سیستم و سپس مجموعه‌ای از درخواست‌ها باشد. هر درخواست ورودی نیز دارای پارامترهایی که یک سفارش ورودی به SUT را توصیف می‌کند. در آزمون نرم‌افزار توصیه می‌شود که در هر آزمایه تنها یک شرط مورد آزمایش قرار گیرد که به این ترتیب باید هر آزمایه تنها شامل

[1] Frormal semantics

[2] https://github.com/functional-specification-based-testing/haskell-matching-engine

[3] https://en.wikipedia.org/wiki/Metaheuristic

یک درخواست شود اما داشتن مجموعه‌ای از درخواست‌ها که پشت سر هم در یک آزمایه آمده است، باعث تغییر حالت درون سیستم می‌شود و از آن جایی که SUT و توصیف آن به ازای مجموعه‌ای یکسان از درخواست‌ها، در یک حالت قرار می‌گیرند، نه تنها مشکلی حاصل نمی شود بلکه می‌توانیم حالت‌های بیشتر را بررسی کرده و باگ‌های بیشتری را پیدا کنیم.

هر مجموعه آزمایه، بیانگر یک کروموزوم در GA است. در فرآیند تولید آزمایه، هر مجموعه آزمایه حداکثر شامل ۴۰ آزمایه است. هر آزمایه سه بخش دارد: ۱) یک پارامتر که مشخص می‌کند یک آزمایه در یک مجموعه، قابل چشم‌پوشی است یا نه. ۲) تعدادی ویژگی ثابت که وضعیت اولیه سیستم را توصیف می‌کند. ۳) حداکثر ۱۰ درخواست[1] بنابراین با توجه به تعریفی که برای کروموزوم ذکر شد، در صورتی که یک ویژگی در یک سفارش معتبر نباشد، از مجموعه‌ی آزمایه خارج می‌شود.

با توجه به این که مقادیر مختلف برای ویژگی‌های یک درخواست، منجر به تولید درخواست‌های متفاوت و در نتیجه میزان پوشش متفاوت می‌شود. به این ترتیب برای اینکه فضای جستجوی الگوریتم محدود باشد و الگوریتم در نهایت بتواند به یک مجموعه آزمایه‌ی بهینه برسد، برای تعداد آزمایه‌های یک مجموعه آزمایه (که به عنوان اندازه‌ی مجموعه آزمایه از آن نام خواهیم برد.) و تعداد درخواست‌های داخل یک آزمایه، محدودیت وجود دارد. این دو پارامتر بر اساس ویژگی‌های سفارشات در سیستم معاملاتی تعیین شده‌است.

۲-۳-۲- تابع سازواری

همانگونه که پیش از این ذکر شد، برای تعریف تابع سازواری از میزان پوششی که یک مجموعه آزمایه روی مدل می‌دهد، استفاده شده‌است. با وجود این که غالبا پوشش شاخه‌ای[2] به عنوان یک معیار پوشش برتر مورد استفاده قرار می‌گیرد، در این پژوهش، این فرض مورد استفاده قرار نگرفته‌است. زیرا نتایجی که تاثیر پوشش شاخه‌ای را نشان می‌دهند، غالبا حاصل از بررسی آن در زبان‌های رویه‌ای[3] است. برای مثال در زبان‌های تابعی، تفاوت معناداری بین پوشش شاخه‌ای و پوشش عبارت[4] وجود ندارد [4] و علت آن تفاوت‌های ذاتی زبان‌های

[1] درخواست‌های ورودی به سیستم شامل سه نوع سفارش (سفارش جدید، تغییر یک سفارش و کنسل کردن یک سفارش) می‌باشد.

[2] Branch Coverage

[3] procedural

[4] Expression Coverage

تابعی و رویه‌ای است. سبب بنابراین از پوشش‌های عبارت، جایگزین[1] و بولی که روش‌هایی برای زبان‌های تابعی هستند، استفاده شده‌است.

یک برنامه‌ی هسکل را می‌توان به صورت مجموعه‌ای از عبارت‌ها مورد بررسی قرار داد، به طوری که هر عبارت تشکیل شده از عبارت‌های کوچک‌تری باشد. با توجه به این که این تعریف کل statementهای برنامه را شامل می‌شود، پوشش عبارت می‌تواند داده‌ی مناسبی برای بررسی میزان پوشش برنامه باشد. تعداد عبارت‌های ارزیابی‌شده با توجه به تعداد کل عبارت‌ها، تعریف تابع سازواری برای پوشش عبارت است. هر عبارت، هنگامی به عنوان یک عبارت ارزیابی‌شده در نظر گرفته می‌شود که حداقل یکی از عبارت‌های زیرمجموعه‌ی آن بررسی شده‌باشد.

عبارت بولی در هسکل به سه شکل است؛ شرط‌ها، guards و qualifiers. Guard یک عبارت بولی است که با عبارت دیگری در ارتباط است؛ هنگامی که نتیجه‌ی ارزیابی Guard، true باشد، عبارتی که با آن در ارتباط است، اجرا می‌شود. شرط‌ها هم مانند جملات شرطی هستند که در زبان‌های رویه‌ای وجود دارد. Qualifier هم حالت کلی Guard است؛ به طوری که به ازای نتایج مختلف حاصل از ارزیابی عبارت بولی، عبارت‌های مختلفی اجرا می شود. پوشش بولی میزان هر یک از مقادیر true و false که توسط هر زمینه بولی syntactic ارزیابی شده را اندازه‌گیری می‌کند. تعداد عبارت‌های بولی که نتیجه‌ی ارزیابی آن‌ها هر دو مقدار true و false بوده‌است، با توجه به تعداد کل عبارت‌های بولی موجود در برنامه، میزان پوشش بولی در تابع سازواری را تعریف می‌کند.

مفهوم جایگزین در یک برنامه‌ی هسکل، به عنوان یک clause در یک تابع یا شاخه در الگوی ME تعریف می‌شود. تعداد جایگزین‌های ارزیابی‌شده با توجه به تعداد کل جایگزین‌ها، میزان پوشش جایگزین در تابع سازواری را تعریف می‌کند.

سازواری می‌تواند به صورت $C(T)$ که به معنای میزان پوشش روی مدل به ازای مجموعه آزمایه‌ی T است، باشد. علاوه بر این، در روش تولید آزمایه‌ی جستجو محور، اندازه‌ی مجموعه‌ی آزمایه نیز می‌تواند یک پارامتر برای تعیین محدودیت عمق جستجو باشد و سازواری به صورت $k_1C(T) - k_2S(T)$ نیز قابل تعریف است. به طوری که $S(T)$ به معنی اندازه مجموعه آزمایه تعداد آزمایه‌های داخل مجموعه) و k_1 و k_2

[1] Altervative Coverage

پارامترهایی برای نرمال‌سازی دو پارامتر هستند. لازم به ذکر است که ابزار مورد استفاده برای اندازه‌گیری پوششی که یک مجموعه آزمایه می‌دهد، hpc است. [5]

۳-۳-۲- جمعت اولیه و عملگرهای جستجو در GA

در GA، از عملگرهای استاندارد crossover[1] و mutation[2] استفاده شده‌است. عملگر crossover دو فرزند O_1 و O_2 را از دو والد (مجموعه آزمایه) P_1 و P_2 تولید می‌کند. هر والد به قسمت تقسیم می‌شود و مثلا O_1 از نیمه‌ی اول P_1 و نیمه‌ی دوم P_2 تولید شده‌است. با توجه به مستقل بودن آزمایه‌ها از هم، خروجی اعمال این عملکرد، دو مجموعه‌ی آزمایه‌ی معتبر خواهد بود. عملگر mutation نیز می‌تواند سه نوع تغییر اعمال کند: ۱) یک آزمایه را از مجموعه آزمایه حذف کند. ۲) یک آزمایه را به مجموعه آزمایه اضافه کند. ۳) هر پارامتر در یک آزمایه را جایگزین کند. عملگر crossover با احتمال ۰٫۵ و عملگر mutation با احتمال ۰٫۱ بر یک مجموعه آزمایه اعمال می‌شود. در راند اول اجرای GA، جمعیت اولیه به صورت رندوم تولید می‌شود؛ اما از آن پس در هر بار اجرا، ۳۰٪ بهترین جمعیت برای اجرای بعدی انتخاب می‌شوند و بقیه به صورت رندوم تولید می‌شود.

۴-۲- ارزیابی

نکته‌ی اولیه در ارزیابی و اجرای آزمایه‌های تولیدی روی SUT، ترجمه‌ی آزمایه‌ها و خروجی حاصل از مدل آن‌ها به پروتکل ورودی و خروجی SUT است.[3] پس از اجرای آزمایه‌ها بر بروی SUT نیز می‌توان با مقایسه‌ی خروجی آن و خروجی تولیدی مدل (که خروجی مورد انتظار و درست است.) درستی SUT را بررسی کرد. طی این برر سی‌ها، آزمایه‌هایی تولید شده، کارآمد ارزیابی شدند؛ زیرا اجرای آن‌ها سبب یافتن باگ‌های متعدد در SUT شده‌است؛ باگ‌هایی که آزمون‌های یکه و یا BDD زیادی که پیش از آن روی سیستم انجام شده بود، موفق به تشخیص آن نشده بودند.

[1] https://www.geeksforgeeks.org/crossover-in-genetic-algorithm/

[2] https://en.wikipedia.org/wiki/Mutation_(genetic_algorithm)

[3] https://github.com/functional-specification-based-testing/translator

سپس برای بررسی کیفیت مجموعه آزمایه‌ی تولید شده، از میزان پوشش شاخه‌ای که مجموعه روی SUT می‌دهد، استفاده شد. به این ترتیب با بررسی پوشش شاخه‌ای حاصل از توابع سازواری مختلف می‌توان آن‌ها را مقایسه کرده و تاثیر اندازه مجموعه آزمایه را نیز مشاهده کرد. با توجه به این که مدلی که از سیستم با زبان هسکل توصیف شده‌است، شامل کل منطق SUT نمی‌شود، تنها چندین کلاس به منظور مقایسه‌ی میزان پوشش انتخاب شده‌است. این انتخاب بر پایه‌ی قابلیت و کارایی کلاس‌ها و همچنین میزان کارایی از آن‌ها که توصیف شده‌است، می‌باشد.

نتایج مقایسه‌ی میزان پوشش حاصل آزمون‌های تیم توسعه و میزان پوشش حاصل از آزمایه‌های تولید شده توسط GA، نشان از نزدیکی این مقادیر داشتند و حتی در برخی موارد میزان پوشش حاصل از آزمایه‌های حاصل از GA بسیار بیشتر بود. سازواری مورد استفاده در این بررسی تنها وابسته به میزان پوشش روی مدل بوده‌است.

سپس با توجه به فرضیاتی که درباره‌ی تاثیر اندازه‌ی مجموعه آزمایه روی عملکرد تابع سازواری وجود داشت، بررسی کوتاهی نیز انجام شد و سازواری با $k_1C(T) - k_2S(T)$ مورد آزمون قرار گرفت و پارامترهای نرمال سازی به گونه‌ای بود که تاثیر میزان پوشش و اندازه‌ی آزمایه برابر باشد. نتایج حاصل نشان‌دهنده‌ی این بود که ممکن است افزودن تاثیر منفی اندازه‌ی مجموعه آزمایه در عملکرد تابع سازواری بهبود ایجاد کند ولی این امر نیازمند بررسی و یافتن وزنی بهینه برای میزان تاثیر اندازه‌ی مجموعه آزمایه می‌باشد.

فصل ۳: جمع‌آوری داده‌ها و نتیجه‌گیری

در این فصل ویژگی‌ها و میزان پوشش حاصل از مجموعه آزمایه‌ها با ضرایب تاثیر متفاوت برای اندازه مجموعه آزمایه به دست‌آمده و سپس داده‌های حاصل مقایسه شده‌اند. در نهایت نیز نتیجه‌ی حاصل بیان شده‌است.

۱-۳- ضریب تاثیر اندازه‌ی مجموعه آزمایه

۱-۱-۳- مقدمه

همان گونه که در بخش ۲-۳-۲ ذکر شد، با قرار دادن سازواری برابر عبارت $k_1C(T) - k_2S(T)$ می‌توان تاثیر اندازه آزمایه را اعمال کرد. با توجه به این که k_1 و k_2 ضرایبی برای نرمالایز کردن دو مقدار هستند، $C(T)$ بر حسب درصد و به عبارتی مقداری بین صفر تا صد است و $S(T)$ نیز می‌تواند عددی بین صفر تا ۴۰ باشد؛ مقدار k_1 برابر $\frac{1}{100}$ و مقدار k_2 برابر $\frac{1}{40}$ می‌باشد. سپس با توجه به این که می‌خواهیم ضرایب تاثیر مختلف اندازه آزمایه را بررسی کنیم، فرمول را به $(\frac{1}{100})C(T) - (\frac{1}{40})(w)S(T)$ تغییر می‌دهیم به طوری که w عددی بین صفر تا یک است. باید توجه داشت که میزان پوشش و اندازه مجموعه آزمایه، اعدادی مثبت هستند و از طرفی هدف غایی حداکثر کردن پوشش و نه حداقل کردن اندازه آزمایه است. بنابراین ضریب بیشتر از یک برای اندازه مجموعه آزمایه، از اساس بی‌معناست و ما تنها می‌خواهیم با پیدا کردن ضریبی بهینه، تابع سازواری را بهبود ببخشیم. به این منظور سعی می‌کنیم با قرار دادن مقادیر مختلف به جای w به نتیجه برسیم. برای $C(T)$ نیز از معیار پوشش بولی استفاده شده‌است. به عبارت دیگر تابع سزواری وابسته به ترکیب میزان پوشش بولی و اندازه مجموعه آزمایه می‌باشد.

برای گزارش میزان پوششی که بر روی SUT به دست آمده، تعداد معدودی کلاس بر حسب عملکرد آن‌ها و میزان منطقی از آن‌ها که توسط مدل توصیف شده‌است، به کمک پژوهشگران انتخاب شده‌است.

لازم به ذکر است که برای قابل اتکا بودن نتایج، به ازای هر w، کل الگوریتم ۵ بار اجرا شده (۵ سری نتیجه تولید می‌شود.) و تمامی اعداد گزارش شده، میانگین مقدار آن‌ها به همراه انحراف معیار طی این ۵ بار اجرا است.

۲-۱-۳- اعمال ضرایب مختلف

۱-۲-۱-۳- ($w = 0$)

پوشش حاصل از مجموعه آزمایه روی مدل			زمان اجرا (دقیقه)	اندازه مجموعه آزمایه
عبارت	بولی	جایگزین		
78.75 ± 1.78	50.00 ± 0.70	70.25 ± 1.29	260 ± 10	21.4 ± 1.51

جدول 1. نتایج حاصل از وزن 0 اندازه مجموعه آزمایه در سازواری

کلاس هدف	میزان پوشش شاخه‌ای
ContinuesTradingMatchingEngine	79.20 ± 4.40
Order	60.00 ± 0.00
LimitOrder	100.00 ± 0.00
IcebergOrder	66.00 ± 7.66
OrderBook	57.00 ± 6.26
QueueItem	33.00 ± 0.00
IcebergQueueItem	100.00 ± 0.00
CreditObserver	80.60 ± 2.93
PercentageOwnershipObserver	90.00 ± 0.00

جدول 2. میزان پوشش حاصل از وزن 0 اندازه‌ی مجموعه آزمایه در سازواری، روی کلاس‌های منتخب SUT

۳-۱-۲-۲ $(\boldsymbol{w = 0.1})$

پوشش حاصل از مجموعه آزمایه روی مدل			زمان اجرا (دقیقه)	اندازه مجموعه آزمایه
عبارت	بولی	جایگزین		
76.60 ± 1.74	48.60 ± 0.48	68.40 ± 1.35	198 ± 10	15.0 ± 2.44

جدول 3. نتایج حاصل از وزن 0.1 اندازه مجموعه آزمایه در سازواری

کلاس هدف	میزان پوشش شاخه‌ای
ContinuesTradingMatchingEngine	76.00 ± 7.23
Order	58.00 ± 4.00
LimitOrder	70.00 ± 24.49
IcebergOrder	62.40 ± 10.51
OrderBook	51.40 ± 2.80
QueueItem	33.00 ± 0.00
IcebergQueueItem	93.20 ± 13.60
CreditObserver	78.60 ± 5.38
PercentageOwnershipObserver	90.00 ± 0.00

جدول 4. میزان پوشش حاصل از وزن .10 اندازه‌ی مجموعه آزمایه در سازواری، روی کلاس‌های منتخب SUT

۳-۱-۲-۳ $(\boldsymbol{w = 0.3})$

اندازه مجموعه آزمایه	زمان اجرا (دقیقه)	پوشش حاصل از مجموعه آزمایه روی مدل		
		جایگزین	بولی	عبارت
8.8 ± 2.77	171 ± 6	66.20 ± 0.48	45.20 ± 1.16	75.20 ± 1.32

جدول 5. نتایج حاصل از وزن 0.3 اندازه مجموعه آزمایه در سازواری

کلاس هدف	میزان پوشش شاخه‌ای
ContinuesTradingMatchingEngine	62.80 ± 16.15
Order	52.00 ± 16.00
LimitOrder	80.00 ± 40.00
IcebergOrder	47.20 ± 14.78
OrderBook	47.00 ± 9.87
QueueItem	29.60 ± 6.80
IcebergQueueItem	86.60 ± 26.80
CreditObserver	75.00 ± 10.35
PercentageOwnershipObserver	85.20 ± 9.60

جدول 6. میزان پوشش حاصل از وزن .30 اندازه‌ی مجموعه آزمایه در سازواری، روی کلاس‌های منتخب SUT

$(w = 0.5)$ -۳-۱-۲-۴

اندازه مجموعه آزمایه	زمان اجرا (دقیقه)	پوشش حاصل از مجموعه آزمایه روی مدل		
		جایگزین	بولی	عبارت
5.8 ± 2.38	160 ± 8	66.20 ± 1.83	44.80 ± 3.60	74.40 ± 2.41

جدول 7. نتایج حاصل از وزن 0.5 اندازه مجموعه آزمایه در سازواری

کلاس هدف	میزان پوشش شاخه‌ای
ContinuesTradingMatchingEngine	67.40 ± 9.37
Order	54.00 ± 4.89
LimitOrder	80.00 ± 24.49
IcebergOrder	59.40 ± 9.32
OrderBook	52.80 ± 5.60
QueueItem	33.00 ± 0.00
IcebergQueueItem	93.20 ± 13.60
CreditObserver	76.20 ± 8.05
PercentageOwnershipObserver	86.00 ± 8.00

جدول 8. میزان پوشش حاصل از وزن .50 اندازه‌ی مجموعه آزمایه در سازواری، روی کلاس‌های منتخب SUT

$(w = 0.8)$ -۳-۱-۲-۵

پوشش حاصل از مجموعه آزمایه روی مدل			زمان اجرا (دقیقه)	اندازه مجموعه آزمایه
عبارت	بولی	جایگزین		
75.60 ± 1.85	44.20 ± 1.32	67.40 ± 1.01	157 ± 4	5.4 ± 1.51

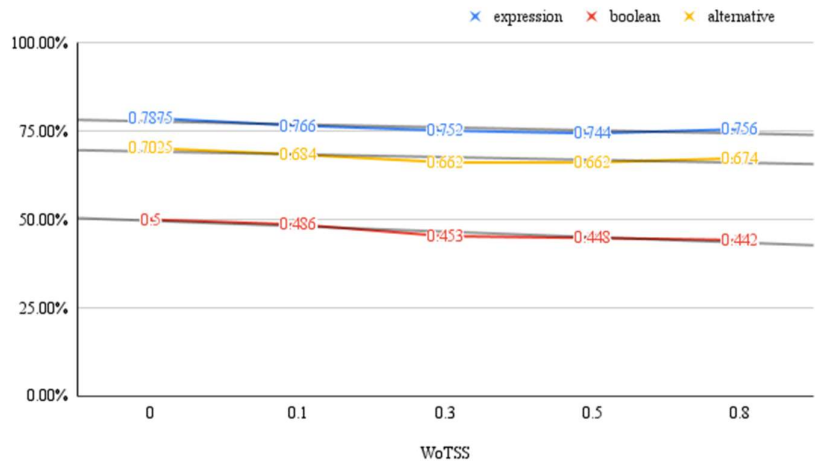
جدول 9. نتایج حاصل از وزن 0.8 اندازه مجموعه آزمایه در سازواری

کلاس هدف	میزان پوشش شاخه‌ای
ContinuesTradingMatchingEngine	68.40 ± 5.53
Order	56.00 ± 4.89
LimitOrder	80.00 ± 24.49
IcebergOrder	60.80 ± 12.15
OrderBook	51.40 ± 2.80
QueueItem	33.00 ± 0.00
IcebergQueueItem	100.00 ± 0.00
CreditObserver	72.80 ± 4.06
PercentageOwnershipObserver	86.00 ± 8.00

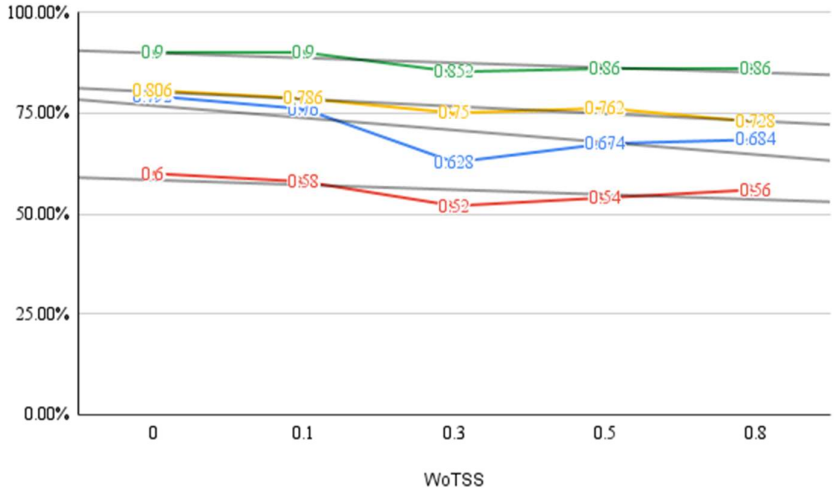
جدول 10. میزان پوشش حاصل از وزن 0.8 اندازه‌ی مجموعه آزمایه در سازواری، روی کلاس‌های منتخب SUT

۳-۱-۳- جمع‌بندی

در ابتدا به منظور مقایسه‌ی نتایج به دست‌آمده به ازای هر w، نمودارهای زیر را رسم می‌کنیم.

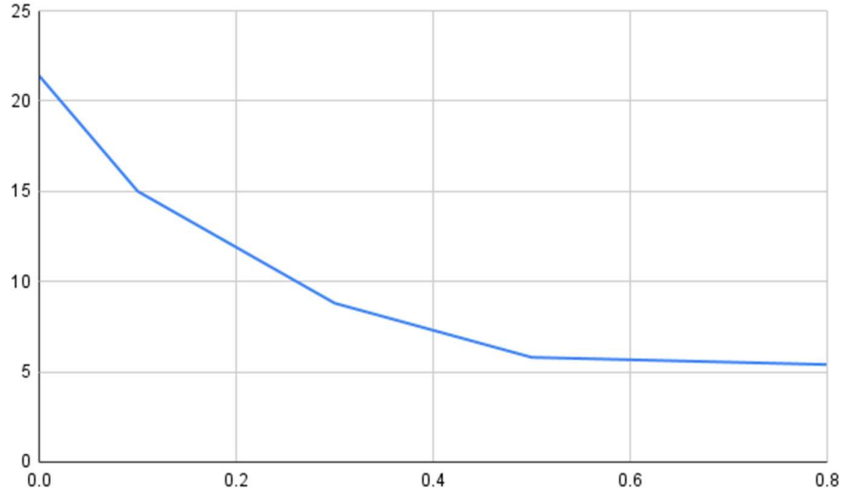


شکل 1. مقایسه‌ی میزان انواع پوشش روی مدل به ازای وزن‌های مختلف اندازه مجموعه آزمایه[1]

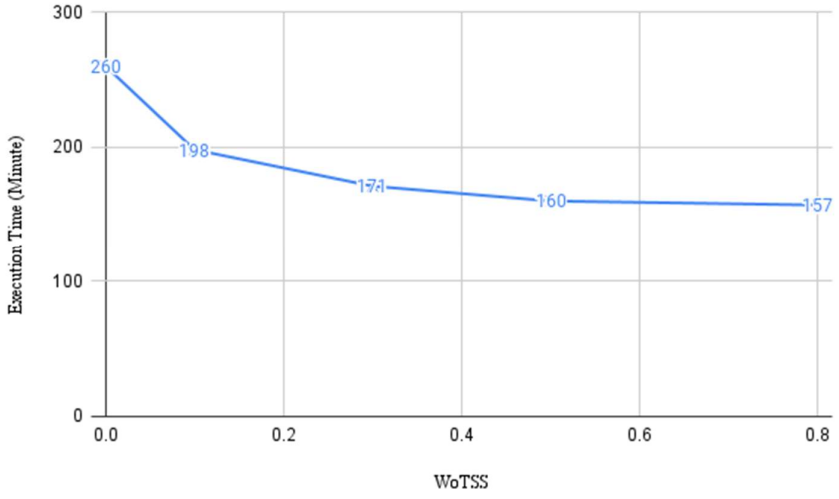


شکل 2. مقایسه‌ی میزان پوشش شاخه‌ای روی چهار کلاس منتخب SUT به ازای وزن‌های مختلف اندازه مجموعه آزمایه

[1] WoTSS: وزن اندازه مجموعه آزمایه (Weight of Test Suit Size)



شکل 3. نمودار اندازه مجموعه آزمایه‌ی خروجی الگوریتم بر حسب وزن اندازه مجموعه آزمایه در سازواری

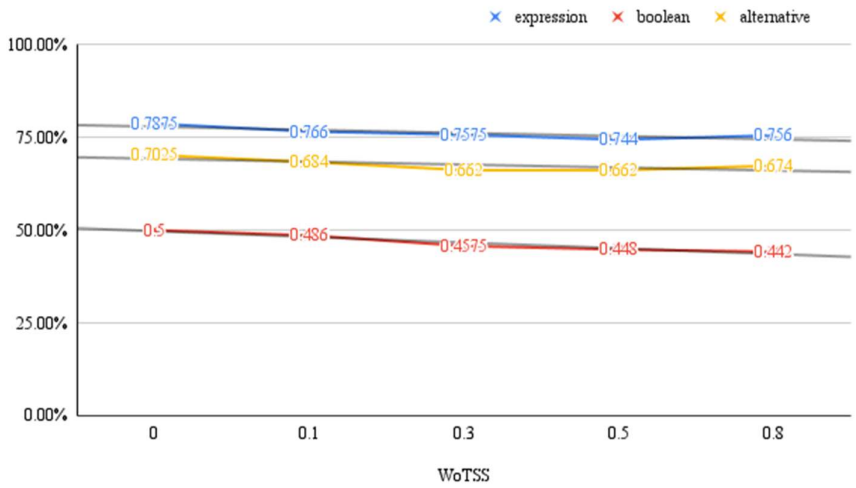


شکل 4. نمودار زمان اجرا بر حسب وزن اندازه مجموعه آزمایه در سازواری

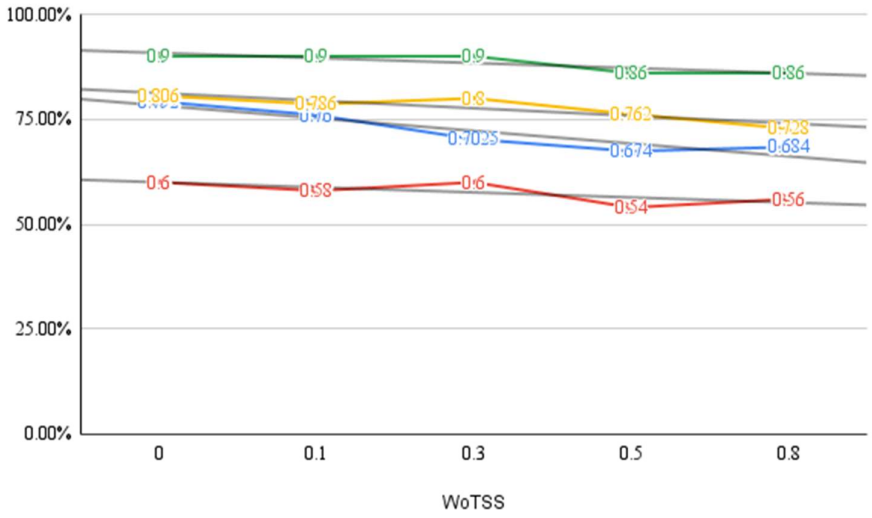
مطابق انتظار با افزایش وزن اندازه مجموعه آزمایه در فرمول سازواری، روند نزولی در پوشش مشاهده می‌شود. هم روی مدل و هم روی SUT، میزان پوشش کاهش می‌یابد. نکته‌ی قابل توجه در این دو نمودار، حداقل بودن مقادیر در وزن ۰٫۳ است. علت این اتفاق، مجموعه آزمایه‌ای است که در دومین اجرا در وزن ۰٫۳ خروجی داده شده‌است. این مجموعه آزمایه، تنها داری ۴ آزمایه است؛ در حالی که مجموعه آزمایه‌های حاصل

از ۴ اجرای دیگر در این وزن، برابر ۱۰، ۱۱، ۱۰ و ۹ است. با مشاهده‌ی انحراف معیار پوشش‌های وزن ۰٫۳ نیز می‌توان متوجه شد که ممکن است داده‌ی خارج از محدوده‌ای وجود داشته باشد. با حذف این خروجی از خروجی‌های وزن ۰٫۳ دوباره نمودارهای بالا را رسم می‌کنیم. نکته‌ی جالب توجه در حذف خروجی خارج از محدود از مجموعه خروجی‌های وزن ۰٫۳، این است که میانگین زمان اجرا تنها یک دقیقه افزایش می‌یابد و رسم دوباره‌ی نمودار آن کمکی نمی‌کند.

سپس دوباره نمودارها را برر سی می‌کنیم؛ با مقای سه‌ی پو شش‌هایی که روی مدل حا صل شده‌ا ست، می‌توان گفت که تغییر پوشش با تغییر وزن اندازه مجموعه آزمایه، ناچیز است. علاوه بر این، میزان پوشش روی SUT برای ما حائز اهمیت اســت. بنابراین در ادامه برای مقایســه و انتخاب وزن ایده‌آل از میزان پوشــش روی SUT استفاده می‌کنیم.

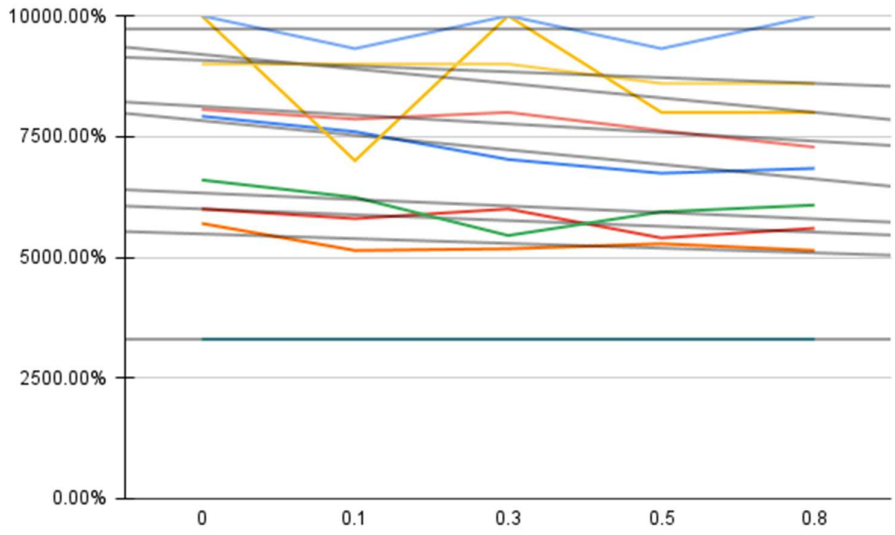


شکل 5. نمودار شکل 1 پس از حذف داده‌ی خارج از محدوده از خروجی‌های وزن 0.3

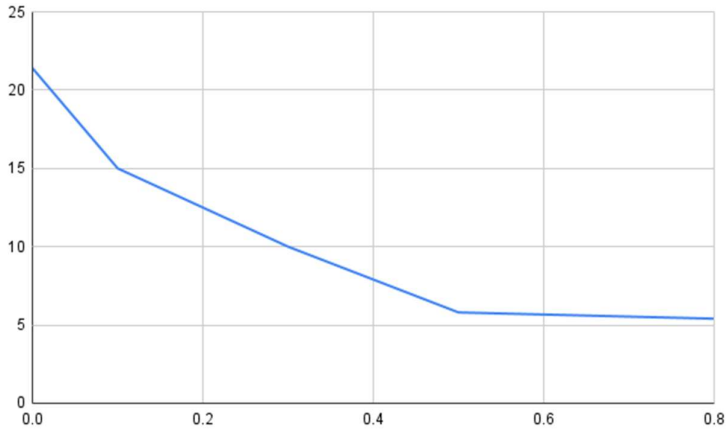


شکل 6. نمودار شکل 2 پس از حذف داده‌ی خارج از محدوده از خروجی‌های وزن 0.3

در نمودارهای بالا، ۴ کلاس SUT از بین ۹ کلاسی که پوشش را برای آن‌ها گزارش کرده‌ایم، استفاده شده‌است. زیرا روند کاهش پوشش در بین کلاس‌های SUT تقریبا مشابه است و ما تنها می‌خواهیم دیدی کلی از آن دا شته با شیم. با این وجود، نمودار پو شش برای هر ۹ کلاس در زیر آمده‌ا ست. م شاهده می شود که روند کلی پوشش روی SUT بجز در برخی موارد که نواسانات کوچکی وجود دارد، نزولی است.



شکل 7. مقایسه‌ی میزان پوشش شاخه‌ای روی کلاس‌های SUT به ازای وزن‌های مختلف اندازه مجموعه آزمایه



شکل 8. نمودار شکل نمودار شکل 3 پس از حذف داده‌ی خارج از محدوده از خروجی‌های وزن 0.3

بدیهی است که اندازه‌ی مجموعه آزمایه نیز کاهش یافته‌است. طبق انتظار زمان اجرای الگوریتم نیز کاهش یافته‌است. نکته‌ی قابل توجه در این مشاهدات کاهش قابل توجه اندازه‌ی مجموعه آزمایه و زمان اجرا در مقابل کاهش میزان پوشش روی SUT می‌باشد. همچنین با توجه به کاهش ناچیز میزان پوشش روی مدل، می‌توان گفت که سازواری با وجود تاثیر منفی اندازه‌ی آزمایه، به مقدار بهینه‌ی خود در حالتی که تنها وابسته به میزان پوشش است، نزدیک شده‌است.

۴-۱-۳- نتیجه‌گیری

همانگونه که در بخش قبلی به آن اشـــاره شـــد، اگرچه میزان پوشـــش روی مدل با افزیش وزن اندازه مجموعه آزمایه در سازواری، کاهش می‌یابد؛ اما میزان کاهش قابل توجه نیست. به عبارتی می‌توان گفت مدل تا نقطه‌ی بهینه‌ی پوششی که پیش از این داشت، فاصله‌ی کوتاهی دارد. در مورد میزان پوشش روی SUT، کاهش، قابل چشم‌پوشی نیست و همانگونه که در نتایج فصل ۲ بیان شد، با افزایش وزن اندازه مجموعه آزمایه به یک، این کاهش چشـــم‌گیر خواهد بود. ســـعی ما بر این اســـت که نقطه‌ای روی نمودار میزان پوشـــش بر روی کلاس‌های SUT بیابیم که با وجود کم بودن زمان اجرا در آن نقطه نسـبت به وزن صـفر، میزان پوشـش تغییر تاثیرگذاری نکرده باشد. با توجه به شکل ۴، شیب کاهش زمان اجرا با افزایش وزن، کاهش می‌یابد و به این ترتیب انتخاب وزنی کمتر از ۰٫۳ برای اندازه مجموعه آزمایه بهینه به نظر می‌رســـد. دوباره نمودار کاهش پوشـــش روی SUT بررســـی می‌کنیم تا بین ۰٫۱ و ۰٫۳ یکی را به عنوان وزن بهینه انتخاب کنیم. اگر اندازه‌ی مجموعه آزمایه به

دلیل زمان اجرای آن روی SUT برای ما حائز اهمیت می‌بود، می‌توانستیم آن را به عنوان پارامتری برای مقایسه‌ی وزن‌ها انتخاب کنیم. ولی از آن جایی که در SUT مسئله‌ای از این بابت نداریم، از این شاخص استفاده نمی‌کنیم. در نهایت با اتکا به میزان پوشش کلاس‌های منتخب و زمان اجرا در دو وزن ۰٫۳ و ۰٫۱ می‌توان گفت مقدار ۰٫۳ برای w، بهینه‌ترین سازواری با فرمول $(\frac{1}{100})C(T) - (\frac{1}{40})(w)S(T)$ را می‌دهد.

۲-۳- ترکیب معیارهای پوشش مختلف (بولی، جایگزین و عبارت)

در ابتدا این فرض وجود داشت که به دلیل رفتارهای متفاوت کلاس‌های SUT، حداکثر کردن یک نوع پوشش در سازواری ممکن است پوشش شاخه‌ای در تعدادی کلاس‌های SUT را بهبود قابل توجه ببخشد. به عبارتی ممکن است پوشش حداکثری روی کلاس‌های مختلف از مجموعه آزمایه‌هایی با سازواری متفاوت به دست آید. به این منظور ابتدا سعی کردیم با ترکیب خطی معیارهای مختلف در سازواری، این امر را بررسی کنیم؛ اما به دلیل عدم درک درست از رابطه و همبستگی معیارهای مختلف و نیازمندی به بررسی‌های کامل‌تر، تلاش‌ها پیرامون این بحث ادامه نیافت و از محدوده‌ی مباحث پروژه خارج شد.

مراجع

مراجع

[1] Zakeriyan A, Khosravi R, Safari H, Khamespanah E, "Towards Automatic Test Case Generation for Industrial Software Systems Based on Functional Specification," in *FSEN 2021*, Tehran, Iran, 2021.

[2] Aichernig BK, "A Testing Perspective on Algebraic, Denotational, and Operational Semantics," *Springer,* pp. 22-38, 2016.

[3] Fraser G, Arcuri A, "Evolutionary Generation of Whole Test Suites," *IEEE Computer Society,* pp. 31-40, 2011.

[4] Cheng Y, Wang M, Xiong Y, Hao D, Zhang L, "Empirical evaluation of test coverage for functional programs," *IEEE,* pp. 2550-265, 2016.

[5] Gill A, Runciman C, "Haskell program coverage," pp. 1-12, 2007.

Abstract:

Because of the complexity of high-capability software systems, it is difficult to generate test cases automatically as they have some non-functional requirements. But we can use a specification-based testing technique to create a model of SUT using functional languages. And generate test cases using search-based testing automatically. It is important to define sufficient fitness function. Fitness function affects the quality of the generated test cases. In this project, the SUT is the stock trading matching engine. In previous researches, matching engine logic was implemented by Haskell and is used as a model for automatic test case generation. We want to improve the fitness function due to its importance. Fitness function depended on source code coverage in previous researches. However, in this project, we investigated the effect of test suite size in fitness and improved the performance of fitness function.

Keywords:
Fitness Function, Size of Test Suite, Coverage, Automatic Test Case Generation

University of Tehran

College of Engineering

School of Electrical and Computer Engineering

Thesis Title

Defining Sufficient Fitness Function for Search-Based Testing in Specification-Based Approach

A thesis submitted to the Undergraduate Studies Office

In partial fulfillment of the requirements for

The degree of bachelor in Computer Engineering (Software)

By:

Seyede Mehrnaz Shamsabadi

Student ID:

810196493

Supervisor:

Dr. Ramtin Khosravi

January 2022