تولید مورد آزمون مبتنی بر مدل از توصیفات تبدیل گراف با استفاده از الگوریتم جستجوی پرتو

مریم عسگری عراقی[۱]، دانشجوی کارشناسی ارشد؛ وحید رافع[۲]، دانشیار؛ اکرم کلائی[۳]، کارشناسی ارشد

۱- دانشکده فنی مهندسی - دانشگاه اراک - اراک - ایران – m-asgariaraghi@msc.araku.ac.ir
۲- دانشکده فنی مهندسی - دانشگاه اراک - اراک - ایران – v-rafe@araku.ac.ir
۳- دانشکده فنی مهندسی - دانشگاه اراک - اراک - ایران – a-kalaee@arshad.araku.ac.ir

چکیده: آزمون نرم‌افزار یکی از فعالیت‌های اساسی در چرخه حیات توسعه نرم‌افزار است که نقش مهمی در کیفیت نرم‌افزار دارد. اغلب بیش از نیمی از هزینه و زمان توسعه نرم‌افزار، صرف آزمون آن می‌شود. بدیهی است که خودکارسازی آزمون و به‌طور ویژه تولید مورد آزمون که از کلیدی‌ترین فعالیت‌های این فرآیند است کمک شایانی در کاهش این هزینه خواهد داشت. آزمون مبتنی بر مدل، از جمله روش‌های موفق خودکارسازی آزمون است که از ابزارهای وارسی مدل نیز برای استخراج موارد آزمون بهره می‌برد. ازآنجاکه این ابزارها در اصل برای وارسی مدل طراحی شده‌اند نه برای تولید آزمون، پژوهش‌های ارائه شده در این زمینه با چالش‌هایی اساسی مانند انفجار فضای حالت و تکراری بودن بخش اعظمی از موارد آزمون روبرو هستند. در پژوهش جاری، راهکاری مبتنی بر الگوریتم جستجوی پرتو ارائه می‌کنیم که از روی توصیفات تبدیل گراف مسئله مجموعه آزمون تولید می‌کند. راهکار پیشنهادی نه‌تنها چالش‌های ذکر شده را بهبود می‌بخشد، بلکه مجموعه آزمونی با پوشش بالا و اندازه کم با صرف بودجه زمانی مطلوب تولید می‌کند. ما آن را در ابزار وارسی مدل GROOVE پیاده‌سازی کرده‌ایم. به‌منظور ارزیابی راهکار پیشنهادی، ما آن را با آزمون مبتنی بر وارسی مدل، راهبردهای مبتنی بر جستجو و آزمون تصادفی مقایسه کرده‌ایم. نتایج آزمایش‌ها روی چندین مطالعه موردی در حوزه سیستم‌های سرویس‌گرا، مؤید برتری روش پیشنهادی از نظر میزان پوشش، اندازه مجموعه آزمون و سرعت است.

واژه‌های کلیدی: آزمون نرم‌افزار، تولید مورد آزمون، آزمون مبتنی بر مدل، الگوریتم جستجوی پرتو، سیستم تبدیل گراف.

Model-Based Test Case Generation from Graph Transformation Specifications using Beam Search Algorithm

M. AsgariAraghi[1], M.sc Student; V. Rafe[2], Associate Professor; A. kalaee[3], M.sc

1- Faculty of Engineering, Arak University, Arak, Iran, Email: m-asgariaraghi@msc.araku.ac.ir
2- Faculty of Engineering, Arak University, Arak, Iran, Email: v-rafe@araku.ac.ir
3- Faculty of Engineering, Arak University, Arak, Iran, Email: a-kalaee@arshad.araku.ac.ir

Abstract: Software testing is one of the key activities in software development life cycle that plays an important role in software quality. More than half of the software development costs and time are often spent on the test. Obviously, the automation of software testing, especially in generating test cases that is a key activity of this process, will dramatically reduce the costs. Among the prosperous testing techniques is model-based testing that utilizes model checker tools to automatically extract test cases. However, as these tools basically designed for model verification, not for test generation, the researches in the testing context are encountered with some major challenges such as state space explosion problem and duplication of the vast majority of test cases. In this paper, we propose a novel method using Beam-search algorithm for generating tests from systems specified through graph transformation specification. The popopsed approach not only improvs the mentioned challenges, but also generates the test suites with high coverage and low size in a desired time budget. We implemented it in the model checker tool GROOVE. To assess the efficiency of our approach, we compared it with model checker-assisted testing, search-based testing strategies and random testing. The empirical results over some case studies in the domain of service-oriented systems confirm it's superiority in terms of coverage size, test suit size and speed.

Keywords: Software testing, test case generation, model-base testing, beam search algorithm, graph transformation system.

تاریخ ارسال مقاله: ۱۳۹۶/۰۵/۱۴
تاریخ اصلاح مقاله: ۱۳۹۶/۰۸/۲۲
تاریخ پذیرش مقاله: ۱۳۹۶/۱۱/۰۹
نام نویسنده مسئول: وحید رافع
نشانی نویسنده مسئول: ایران - اراک - سردشت – پردیس دانشگاه اراک – دانشکده فنی مهندسی - گروه کامپیوتر

۱ -مقدمه

رشد چشمگیر کاربرد نرم‌افزارها در سیستم‌های متنوع، آزمون نرم‌افزار را به یک مرحله مهم و ضروری در چرخه حیات توسعه نرم‌افزار مبدل کرده است [۱]. اما میزان تلاشی که صرف تولید یک مجموعه آزمون با قابلیت بالای شناسایی خطا می‌شود هنوز مسئله‌ای چالش‌برانگیز است. پژوهش‌ها نشان می‌دهند که هزینه کلی آزمون، حدود ۵۰٪ هزینه توسعه است [۲]. ازاین‌رو بدیهی است که خودکارسازی فرآیند آزمون نرم‌افزار و طبیعتاً تولید خودکار مورد آزمون به‌عنوان یک بخش کلیدی از این فرآیند، تأثیر به‌سزایی در بهینه‌سازی فرآیند آزمون خواهد داشت [۳].

روش‌های مختلفی برای طراحی موارد آزمون وجود دارد که می‌توان آن‌ها را به دو روش جعبه سفید (آزمون ساختاری) و جعبه سیاه (آزمون عملکردی) تقسیم‌بندی کرد [۱]. در رویکرد جعبه سفید، کد برنامه برای طراحی آزمون استفاده می‌شود و با مکانیسم داخلی یک سیستم سروکار داریم درحالی‌که در رویکرد جعبه سیاه فقط روی خروجی‌های سیستم تحت آزمون تمرکز می‌شود. این روش که مبتنی بر مشخصات نیازمندی‌ها است نیازی به اجرای کد ندارد و می‌تواند به شناسایی هرگونه ابهام و ناسازگاری در مشخصات نیازمندی‌ها کمک کند.

از جمله راهکارهای جعبه سیاه، می‌توان به آزمون مبتنی بر مدل (MBT)[۱] اشاره کرد که سال‌هاست در پژوهش‌های صنعتی و دانشگاهی مورد استفاده قرار گرفته است [۴]. این راهکار، با تحلیل مدل سیستم تحت آزمون و در نظر گرفتن معیار پوشش[۲]، قادر است به‌صورت سیستماتیک موارد آزمونی تولید کند که ویژگی‌های خاصی از مدل را می‌پوشاند.

وارسی مدل[۳] [۵]، یکی از راهبردهای شناخته‌شده و پرکاربردی است که MBT از آن برای استخراج خودکار دنباله‌های مورد آزمون بهره می‌گیرد. در این روش اهداف آزمون به‌صورت ویژگی‌هایی فرمال بیان می‌شوند و ابزار وارسی مدل با تولید کل فضای حالت به یافتن مسیری جهت اثبات یا نقض آن ویژگی می‌پردازد. چنین مسیری به‌خوبی می‌تواند معرف یک مورد آزمون باشد [۶]. علی‌رغم موفقیت‌ها، از جمله چالش‌های اصلی این ابزارها، مسئله انفجار فضای حالت[۴] است که در آن با افزایش اندازه فضای حالت، وارسی ممکن است بسیار زمان‌بر و یا حتی به دلیل کمبود حافظه غیرممکن باشد [۷]. مشکل دیگر این راهکار تولید آزمون، تکراری بودن درصد زیادی از موارد آزمون تولید شده است. بسیاری از موارد آزمون، پیشوند یکسانی دارند و به میزان پوشش مجموعه آزمون، کمکی نمی‌کنند [۸].

هدف از پژوهش جاری، ارائه راه‌حلی برای بهبود چالش‌های موجود در راهکارهای تولید آزمون مبتنی بر وارسی مدل است. به‌منظور مقابله با مشکل انفجار فضای حالت باید دنبال روش‌هایی باشیم که فضای حالت سیستم را به‌طور هوشمندانه پیمایش می‌کنند. پژوهش‌های ارائه شده [۹، ۱۰] نشان می‌دهند که بهره‌گیری از راهکارهای فرامکاشفه‌ای [۱۱] در وارسی مدل سیستم‌های بزرگ و پیچیده جهت مقابله با مسئله انفجار فضای حالت مؤثر هستند.

ماهیت الگوریتم جستجوی پرتو بدین‌صورت است که محاسبات را کاهش می‌دهد و درنتیجه زمان جستجو نیز کاهش می‌یابد؛ همچنین مصرف حافظه آن نسبت به الگوریتم‌های دیگر جستجو کمتر است [۱۲]. با توجه به این مزایا می‌توان از این الگوریتم در تولید موارد آزمون بهره گرفت زیرا با صرف زمان کمی می‌توان به تولید مجموعه آزمون پرداخت، درحالی‌که الگوریتم‌های جستجوی دیگر به زمان بیشتری نیاز دارند و زمان تولید موارد آزمون یکی از معیارهای ارزیابی روش‌های مختلف تولید موارد آزمون است. همچنین مسئله انفجار فضای حالت را با تکیه بر این موضوع که مصرف حافظه در این الگوریتم نسبت به سایر روش‌ها کمتر است، می‌توان تا حدودی بهبود داد. ازاین‌رو، ما از الگوریتم جستجوی پرتو[۵] جهت هدایت فضای حالت به سمت تولید مجموعه آزمون‌هایی با میزان پوشش بالاتر و اندازه کمینه نسبت به راهکارهای موجود استفاده می‌کنیم. راهکار پیشنهادی از ویژگی‌های مدل‌سازی و شبیه‌سازی ابزارهای وارسی مدل استفاده می‌کند که آن را در ابزار وارسی توصیفات تبدیل گراف GROOVE[۶] [۱۳] پیاده‌سازی کرده‌ایم و با استفاده از چندین مطالعه موردی در حوزه سیستم‌های سرویس‌گرا به بررسی نتایج پرداختیم. همچنین به‌منظور ارزیابی راهکار پیشنهادی الگوریتم‌های تولید مورد آزمون مبتنی بر وارسی مدل و تصادفی نیز پیاده‌سازی شده است. الگوریتم‌ها بر اساس فاکتورهای متعددی مانند میزان پوشش، اندازه مجموعه آزمون و سرعت مقایسه شده‌اند. نتایج آزمایش‌ها مؤید آن است که راهکار پیشنهادی نسبت به راهکارهای تولید آزمون رایج مانند آزمون مبتنی بر جستجو و آزمون مبتنی بر وارسی مدل از نظر میزان پوشش، اندازه مجموعه آزمون و زمان تولید موفق عمل کرده است.

در ادامه در بخش ۲ به معرفی کارهای مرتبط در زمینه تولید مورد آزمون می‌پردازیم. در بخش ۳ تعاریف و مفاهیم بنیادی مقاله را شرح می‌دهیم. بخش ۴ به ارائه راه‌حل پیشنهادی اختصاص دارد. در بخش ۵ نتایج و ارزیابی روش پیشنهادی نسبت به سایر راهکارهای موجود آمده است. در بخش ۶ نتیجه‌گیری و نیز پیشنهاداتی برای کارهای آتی مطرح می‌شود.

۲ -کارهای مرتبط

الگوریتم جستجوی پرتو را می‌توان برای تولید بهینه موارد آزمون مبتنی بر کد استفاده کرد. برای نمونه [14] از الگوریتم جستجوی پرتو به همراه یک جدول راهنما که شامل داده‌های خاصی از اجرای قبلی است استفاده می‌کند. این روش فضای حالت را کاراتر از روش‌های موجود جستجو می‌کند و نتایج نیز نشان می‌دهند که می‌تواند میزان پوشش را بیش از ۲۰٪ بهبود دهد.

با بهینه‌سازی ترتیب اجرای موارد آزمون، کارایی آزمون نرم‌افزار بالا می‌رود، بنابراین شیوه‌های اولویت‌بندی نقش مؤثری در آزمون نرم‌افزار دارند. در پژوهش [۱۵] الگوریتم جستجوی پرتو محلی برای اولویت‌بندی موارد آزمون استفاده شده است که این راهکار مجموعه‌ای از موارد آزمون را دریافت کرده و با استفاده از معیارهایی، دنباله‌ای از موارد آزمون اولویت‌بندی شده را تولید می‌کند.

در پژوهش [۱۶] از توصیفات تبدیل گراف به‌منظور تولید مورد آزمون بهره گرفته است که به‌منظور بهینه‌سازی فرآیند تولید مورد آزمون و هدایت هوشمند فضای حالت از الگوریتم‌های فرامکاشفه‌ای مانند الگوریتم خفاش (BAT) و الگوریتم جستجوی نیروی گرانشی (GSA)[۷] استفاده کرده است. در بخش ارزیابی به مقایسه روش پیشنهادی با این پژوهش خواهیم پرداخت. تا جایی که می‌دانیم جز پژوهش [۱۶] هنوز کاری در زمینه تولید مورد آزمون ارائه نشده است که به‌صورت مستقیم از توصیفات تبدیل گراف استفاده کند.

یکی از مدل‌هایی که نحو آن مشابه این توصیفات است و تلاش‌هایی برای تولید آزمون از آن صورت گرفته، مدل توافق گرافیکی است که جزء مدل‌های UML [۸] محسوب می‌شود. به‌عنوان مثال در [۱۷] از مدل توافق گرافیکی برای تولید مورد آزمون از وب سرویس‌ها استفاده شده است. در این روش که از ابزار تبدیل گراف AGG[۹] استفاده می‌کند، با دریافت عملیات سرویس‌ها در قالب توافق‌های گرافیکی، وابستگی و تداخل میان آن‌ها به‌صورت ایستا محاسبه می‌شود. سپس با پیمایش خود گراف وابستگی و تداخل، ابتدا دنباله‌هایی از فراخوانی قوانین تولید می‌شود و پس از آن با توجه به حالت اولیه داده شده، اگر قابل اجرا باشند به‌عنوان یک مورد آزمون گزارش می‌شوند. در طی بررسی اجراپذیر بودن این دنباله‌ها، هر یالی از گراف وابستگی که برای اولین بار پیمایش می‌شود، ذخیره شده و سپس به‌عنوان میزان پوشش آن دنباله گزارش می‌شود. در بخش ارزیابی این کار مشخص نیست موارد آزمون تولید شده به چه میزان، معیارهای در نظر گرفته شده از مدل را پوشش داده‌اند. به‌علاوه، در مورد کارایی روش هم بحثی صورت نگرفته است. چون در این روش دنباله‌ها به‌صورت ایستا تولید می‌شوند، به دست آوردن یک دنباله اجراپذیر که یک معیار وابستگی و یا مخصوصاً یک معیار تداخل را به این شیوه پوشش دهد کار بسیار مشکلی است؛ ازاین رو، این امکان وجود دارد که موارد آزمون تولید شده با این روش، پوشش مناسبی از معیارها را فراهم نکنند.

در [۱۸] نیز از توافق‌های گرافیکی برای تولید مورد آزمون استفاده شده است؛ در ابتدا آن‌ها را به زبان برنامه‌ریزی تعریف دامنه (PDDL)[۱۰] تبدیل و سپس از ابزارهای برنامه‌ریزی برای تولید موارد آزمون بهره گرفته‌اند. ورودی این ابزارها متنی است و بر مبنای گزاره‌های منطقی کار می‌کنند. با دریافت مشخصات دامنه مسئله، حالت شروع و یک هدف، این ابزارها به‌صورت مکاشفه‌ای دنباله‌ای از فراخوانی‌ها را که به آن هدف منتهی می‌شوند برمی‌گردانند. در اینجا، هدف پیدا کردن یک دنباله فراخوانی از حالت اولیه سیستم به ازای هر یک از توافق‌ها است. نتیجه ارزیابی این بوده است که استفاده از راهکارهای مکاشفه‌ای این امکان را می‌دهد که بدون تولید کامل فضای حالت، فقط بخشی از آن را تولید کنند و به این شیوه می‌توان مسئله انفجار فضای حالت که در وارسی مدل رخ می‌دهد و مانع از تولید مورد آزمون است را بهبود بخشید. در این کار نیز معیاری برای انتخاب موارد آزمون ارائه نشده است؛ به‌علاوه ارزیابی مناسبی نیز صورت نگرفته است.

در زمینه تولید مورد آزمون با استفاده از معیار پوشش جریان داده کارهای متعددی صورت گرفته است که بیشتر آن‌ها از کد نرم‌افزار برای استخراج موارد آزمون استفاده می‌کنند. در پژوهش [۱۹] برای تولید خودکار موارد آزمون از الگوریتم رقابت استعماری استفاده کرده است که در این پژوهش تابع شایستگی جدیدی طراحی شده است که این تابع در واقع میزان پوشش مسیرهای غیرتکراری گراف جریان کنترلی نرم‌افزار را مورد توجه قرار می‌دهد. نتایج نشان می‌دهد که راهکار پیشنهادی آن‌ها در مقایسه با الگوریتم‌های دیگر مانند الگوریتم ژنتیک و الگوریتم پرندگان و همچنین در مقایسه با استفاده از تابع شایستگی‌های دیگر موفق عمل کرده است.

پژوهش [۲۰] نیز رویکرد جدیدی برای تولید موارد آزمون در سیستم‌های شئ‌گرا برای آزمون جامعیت ارائه می‌دهد. کارهای کمی در زمینه آزمون جامعیت به خاطر مشکل انفجار فضای حالت انجام شده است. در پژوهش ارائه شده از تحلیل تعریف استفاده به‌منظور کاهش فضای جستجو استفاده می‌کند و در نهایت توالی از متدها را تولید می‌کند. درخت آزمون براساس مسیرهای تعریف استفاده گسترش می‌یابد. این راهکار در مقایسه با آزمون تصادفی عملکرد بهتری دارد، زیرا در آزمون تصادفی طول موارد آزمون باید ثابت در نظر گرفته شود درحالی‌که طول موارد آزمون با توجه به نمایش درختی می‌تواند متغیر باشد و به اهداف پوشش بستگی دارد.

در پژوهش [۲۱] از الگوریتم پرندگان برای تولید خودکار موارد آزمون استفاده شده است که در آن معیار جریان داده برای سنجش موارد آزمون به‌کار گرفته شده است. درنهایت راهکار خود را با الگوریتم ژنتیک مقایسه می‌کند و نتایج نشان می‌دهد که الگوریتم پرندگان با تولید تعداد نسل کمتری به میزان پوشش بیشتری می‌رسد.

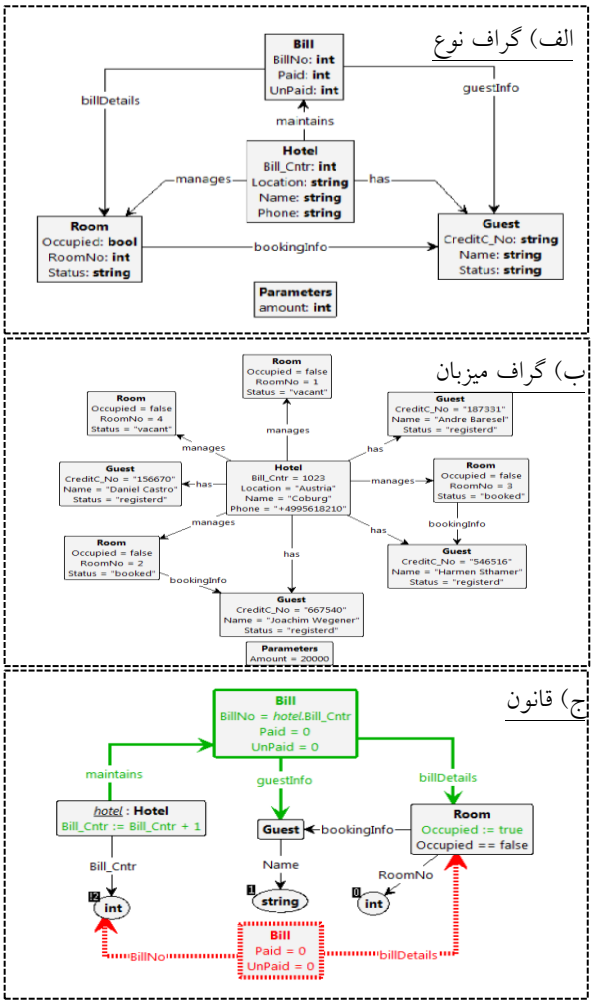
۳ تعاریف و مفاهیم بنیادی

۳ ـ ۱ توصیفات تبدیل گراف

سیستم تبدیل گراف (GTS)[۱۱] [۲۲، ۲۳] یک زبان مدل‌سازی ریاضی‌وار است که در توصیف پیکربندی و رفتار سیستم‌های پویا قابلیت‌های مناسبی دارد. توصیفات تبدیل گراف، با یک ۳ تایی تعریف می‌شوند: GTS = (TG, HG,R) که در آن TG گراف نوع[۱۲]، HG گراف میزبان[۱۳] و R [۱۴] نیز مجموعه‌ای از قوانین است [۲۴].

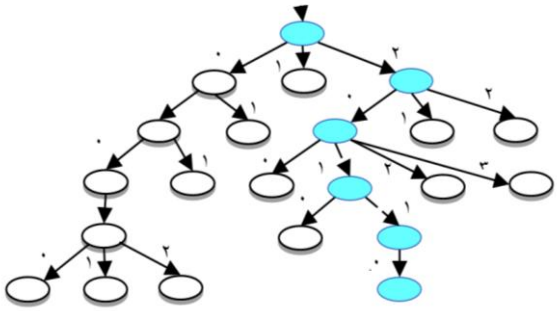
- گراف نوع، یک نمایش کلی از سیستم است و به‌عنوان فرامدل نیز شناخته می‌شود.
- گراف میزبان، نشان‌دهنده وضعیت اولیه سیستم است و باید نمونه‌ای از گراف نوع باشد.
- یک قانون تبدیل، از دو گراف که هر دو هم‌ریخت گراف نوع هستند تشکیل می‌شود. یکی مربوط به بخش سمت چپ (LHS)[۱۵] و دیگری مربوط به بخش سمت راست (RHS)[۱۶] است که با گراف سمت چپ (به‌صورت جزئی) همپوشانی دارد. یک قانون ممکن است دارای شرایط کاربرد منفی (NAC)[۱۷] باشد که در واقع اجرای یک قانون را محدود می‌سازد.

به‌عنوان مثالی از یک سیستم تبدیل گراف، سرویس مدیریت مهمانان هتل را در نظر بگیرید. مهمانی که ثبت‌نام کرده است، می‌تواند یک اتاق موجود را رزرو کند. وقتی مهمان اطلاع می‌دهد که قصد تصفیه حساب دارد به‌صورت خودکار، مبلغ صورت حساب او کسر می‌شود. شکل ۱ گراف نوع، میزبان و یکی از قوانین این سیستم را نشان می‌دهد که در ابزار GROOVE، مدل‌سازی شده است.



شکل ۱: سیستم تبدیل گراف سرویس مدیریت مهمانان هتل.

در یک سیستم تبدیل گراف با اجرای هر قانون، حالت جدیدی ایجاد می‌شود که نشان‌دهنده وضعیت جدید سیستم است. فضای حالت، شامل مجموعه‌ای از این حالات و انتقال بین آن‌ها است. برچسب هر انتقال (گذر خروجی از یک حالت) نام قانون مربوطه است و به هرکدام یک شماره نیز اختصاص می‌یابد. شکل ۲ یک فضای حالت فرضی را نشان می‌دهد (نام قوانین نشان داده نشده است). در این شکل، دنباله شماره گذر قوانین (از سمت چپ) [۲، ۰، ۱، ۱، ۰] مسیری است به طول ۵ که به‌صورت رنگی مشخص شده است. جدول ۱ نیز دنباله فراخوانی قوانین معادل با این مسیر را نشان می‌دهد. مشاهده می‌شود که قوانین می‌توانند پارامتر ورودی داشته باشند که مقادیر آن‌ها با اجرای قوانین و یا از گراف میزبان تعیین می‌شود.



شکل ۲: یک فضای حالت فرضی.

جدول ۱ : دنباله فراخوانی قوانین معادل با مسیر رنگی در شکل (۲).

شماره گذر	فراخوانی قانون
۲	BookRoom(1, “Joachim Wegener”)
۰	OccupyRoom(1, “Joachim Wegener”, 1023)
۱	UpdagteBill(1023, 200000)
۱	ClearBill(1023)
۰	Checkout(1, “Joachim Wegener”, 1023)

۲ ۳ -معیار پوشش جریان داده

ازآنجاکه تعداد موارد آزمونی که از سیستم‌های نرم‌افزاری قابل استخراج است عموماً بسیار زیاد است و هزینه اجرای تمام آن‌ها بالا است، می‌توان معیارهایی تعریف کرد و صرفاً به تولید موارد آزمونی پرداخت که این معیارها را پوشش می‌دهند.

یکی از کاراترین معیارهای پوشش، معیار پوشش جریان داده است [۲۵]. این معیار، یک مجموعه آزمون را در صورتی مناسب می‌داند که در گراف کنترل جریان مسیری از تعریف یک متغیر تا استفاده از آن متغیر امتداد داشته باشد به‌نحوی‌که در این مسیر تعریف مجددی از آن متغیر نباشد (مسیر عاری از تعریف). معیار پوشش جریان داده از این نکته برگرفته شده است که در برنامه‌های نرم‌افزاری تعداد زیادی متغیر تعریف می‌شوند و اگر مقدار یک متغیر بد محاسبه شود تنها زمانی آشکار می‌شود که از آن در جایی از برنامه استفاده شود. تحلیل جریان داده به رابطه بین تعریف متغیرها (def)[۱۸] و استفاده از آن‌ها

(use) می‌پردازد که همان روابط تعریف استفاده (def-use) نامیده می‌شود [۲۵].

در توصیفات تبدیل گراف، قوانین سیستم ممکن است با یکدیگر رابطه وابستگی یا تداخل داشته باشند. فرض کنید R1 و R2 دو قانون از یک سیستم تبدیل گراف باشند. تعاریف ۱ و ۲ شرایط مربوط به وابستگی یا تداخل هر زوج قانون را نشان می‌دهند.

تعریف ۱. با برقراری هر یک از شرایط زیر می‌گوییم قانون R1 به قانون R2 وابسته است و این وابستگی را به‌صورت R2≻R1 نمایش می‌دهیم [۱۷]:

- حداقل یک یال یا گره از LHS قانون R1، توسط RHS قانون R2 اضافه شود.
- حداقل یک یال یا گره از NAC قانون R1، توسط RHS قانون R2 حذف شود.

تعریف ۲. با برقراری هر یک از شرایط زیر می‌گوییم قانون R1 با قانون R2 تداخل دارد و آن را به‌صورت R1↗R2 نمایش می‌دهیم [۱۷]:

- حداقل یک یال یا گره از LHS قانون R1، توسط RHS قانون R2 حذف شود.
- حداقل یک یال یا گره از NAC قانون R1، توسط RHS قانون R2 اضافه شود.

شکل ۳ گراف جریان داده مربوط به مثال مدیریت مهمانان هتل را نشان می‌دهد. در این گراف، گره‌ها همان نام قوانین هستند و یال‌ها نیز روابط وابستگی (با نماد ≻) یا تداخل (با نماد ↗) بین آن‌ها را نشان می‌دهند. به‌علاوه، در ابتدا و انتهای هر یال برچسبی (R : خواندن (Read)، C : ایجاد (Create)، U : به‌روزرسانی (Update) ، D : حذف (Delete)) قرار دارد که نحوه مشارکت گره‌های متصل به این یال را نشان می‌دهد. برای نمونه، قانون OccupyRoom به قانون bookRoom وابسته است به این صورت که قانون bookRoom حداقل یک یال یا گره را ایجاد می‌کند که قانون OccupyRoom برای اجرا شدن آن را می‌خواند.

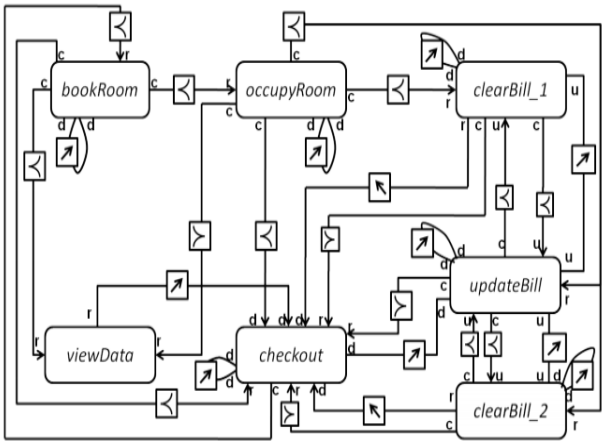
پوشش یال‌های گراف جریان داده می‌تواند به‌عنوان معیار پوشش در تولید مورد آزمون در نظر گرفته شود. جدول ۲ فهرست معیارهایی که در این مقاله در نظر گرفته‌ایم را نشان می‌دهد. این معیارها روابط وابستگی بین قوانین تبدیل گراف را پوشش می‌دهند.

با توجه به آنچه در بالا گفتیم، به کمک معیارهای جریان داده می‌توانیم فهرست اهداف آزمون مسئله را استخراج کنیم. مثلاً در سیستم مدیریت مهمانان هتل اگر معیار پوشش جریان داده معیار شماره ۱ باشد، زوج قانون (bookroom, occupyRoom) یک هدف آزمون است.

۳ ۳ -الگوریتم جستجوی پرتو

الگوریتم جستجوی پرتو [۲۶] یکی از الگوریتم‌های فرامکاشفه‌ای است که گراف را با استفاده از رأس‌هایی که در یک مجموعه خاص، احتمال وجود بیشتری دارند می‌پیماید. در واقع این جستجو حالت بهینه الگوریتم جستجوی اول سطح[۱۹] است که حافظه مورد نیاز را کاهش می‌دهد.

الگوریتم جستجوی اول سطح، پیدا کردن کمترین فاصله بین رأس شروع و پایان را در گراف بی‌وزن تضمین می‌کند ولی برای جستجو در فضاهای بزرگ این کار تقریباً غیرعملی است چرا که حافظه بسیار زیادی مصرف می‌کند و ممکن است قبل از این‌که به جواب برسد حافظه پر شود.



شکل ۳: بخشی از گراف جریان داده مثال سرویس مدیریت مهمانان هتل [۱۶].

جدول۲: انواع معیارهای پوشش توصیفات تبدیل گراف.

معیار	نوع وابستگی
۱	Create-Read
۲	Create-Delete
۳	Create-Update
۴	پوشش تمام وابستگی‌ها (ترکیب معیارهای ۱و ۲و ۳)

الگوریتم پرتو، برای اینکه در حافظه صرفه‌جویی کند یک تابع h برای پیش‌بینی هزینه رسیدن به رأس موردنظر از رأس داده شده در نظر می‌گیرد. همچنین از یک پارامتر به نام k (عرض پرتو) استفاده می‌کند که نشان‌دهنده تعداد رأس‌هایی است که در هر مرحله از الگوریتم جستجوی اول سطح ذخیره شده است. بنابراین زمانی که الگوریتم جستجوی اول سطح همه رأس‌های مرزی را ذخیره می‌کند الگوریتم پرتو فقط k رأس با بهترین مقدار در هر مرحله از جستجو را ذخیره می‌کند. ایده اصلی این است که تابع h به الگوریتم اجازه می‌دهد که رأس‌هایی انتخاب شوند که مسیر را به سمت رأس مقصد راهنمایی کنند و پارامتر k باعث می‌شود که الگوریتم فقط این رأس‌های مهم را ذخیره کند و از پر شدن حافظه قبل از رسیدن به هدف جلوگیری کند. شبه کد الگوریتم جستجوی پرتو در شکل ۴ آمده است.

۳ ۴ -تولید مورد آزمون به کمک وارسی مدل

وارسی مدل یک روش خودکار و دقیق برای تأیید و یا رد سیستم‌های نرم‌افزاری است. این روش با دریافت یک توصیف رسمی از سیستم و نیز یک ویژگی به‌منظور وارسی، تمام حالت‌های ممکن از یک سیستم که فضای حالت نامیده می‌شود را تولید کرده، سپس به‌طور جامع حالات قابل دسترس را به‌منظور وارسی این ویژگی بررسی می‌کند.

```
1.  queue Q = {initial state}
2.  while (Q is not empty)
3.      priority queue Q' = 0
4.      remove S from Q
5.      For each successor state S' of S
6.          If S' not already visited
7.              If S' is the goal
8.                  BREAK
9.              Else
10.                 f ← h(s')
11.                 store (S' , f ) in Q'
12.                 remove all but k best elements from Q'
13.                 Q = Q'
14.             End
15.         End
16.     End
17. End
```

شکل ۴: شبه کد الگوریتم جستجوی پرتو [۲۵].

در وارسی مدل، سیستم و ویژگی‌ها باید با زبان مدل سازی رسمی توصیف شوند (برای مثال منطق زمانی LTL[۲۰]). یکی از مزایای این روش تولید مثال نقض است که برای استخراج خودکار دنباله‌های مورد آزمون می‌توان معیارهای آزمون را در قالب ویژگی‌های منفی تعریف کرد و از توالی ایجاد شده از مثال نقض این ویژگی‌ها برای ساخت دنباله‌های آزمون استفاده کرد [۲۷]. برای نمونه فرض کنید که پوشش زوج $(d^{e}_{rule1}, u^{e}_{rule3})$ یک هدف آزمون مبتنی بر جریان داده است که در آن d حالتی است که با اجرای قانون rule1 به‌دست‌آمده و یال e را ایجاد می‌کند. u حالتی است که با اجرای قانون rule2 به‌دست‌آمده و یال e را استفاده می‌کند. رابطه (۱) یک ویژگی فرمال منفی را برای هدف آزمون فوق با استفاده از منطق زمانی LTL نشان می‌دهد که در آن d و u به ترتیب گره‌هایی از گراف جریان داده هستند که در آن متغیر e تعریف و استفاده می‌شود، G (همیشه)، X (زمان آینده) و U (تا) را نشان می‌دهد.

$$G!(\ d^{e}_{rule1} \wedge X[\sim \mathrm{def}(e)\ U\ u^{e}_{rule3}]) \qquad (۱)$$

این رابطه به این معنی است که در کل فضای حالت هیچ مسیر عاری از تعریفی از گره d به گره u وجود ندارد. اگر وارسی‌کننده مدل مثال نقضی برای آن مشاهده کند مسیر مربوطه را نشان می‌دهد.

متأسفانه، در ابزار GROOVE نمی‌توان یک ویژگی مربوط به جریان داده را وارسی کرد. برای مقایسه رویکرد خود با تولید مورد آزمون مبتنی بر وارسی کننده مدل، مراحل الگوریتم شکل ۵ را اجرا کردیم. این الگوریتم برای هر هدف آزمون که تاکنون توسط مجموعه آزمون پوشش داده نشده با توجه به بودجه جستجو، به دنبال یک مسیر عاری از تعریف (مورد آزمون) می‌گردد. فرض کنید پوشش زوج $(d^{e}_{rule1}, u^{e}_{rule3})$ یک هدف آزمون باشد که این هدف در پارامتر c قرار می‌گیرد. برای این منظور، فضای حالت برای وارسی خصوصیت $G!(\ rule1 \wedge X\ \ rule3)$ که در پارامتر f تعریف شده است، جستجو می‌شود (خط ۶). اگر این ویژگی برآورده شود و یا انفجار فضای حالت رخ دهد هدف آزمون فعلی را به‌عنوان یک هدف غیرقابل آزمون علامت زده و یکی دیگر از اهداف را انتخاب می‌کنیم (خطوط ۷ تا ۹). در غیر این صورت، وارسی‌کننده مدل یک مسیر مثال نقض تولید می‌کند (خط ۱۱). اگر این مسیر با توجه به متغیر e عاری از تعریف باشد به مجموعه آزمون T اضافه می‌شود و هر مورد آزمون t در T که این دنباله تمام اهداف آن را پوشش دهد از T حذف می‌شود (خطوط ۱۲ تا ۲۰). اگر مسیر عاری از تعریف نباشد، تا زمانی که بودجه جستجو موجود باشد جستجوی فضای حالت برای وارسی ویژگی فعلی ادامه می‌یابد. (خطوط ۵ تا ۲۲).

```
Input: Coverage objectives O
Output: Test suite T
1.  MARK all of the test objectives in O as uncovered
2.  Until every test objective in O is marked as covered or untestable
3.      c  CHOOSE an uncovered test objective from O
4.      f CREATE a trap property using LTL formula for c
5.      While search budget
6.          MODEL CHECK f
7.          If f is not satisfy
8.              MARK c as untestable
9.              BREAK
10.         Else
11.             LET p be a counterexample for f
12.             If p is a def-clear path
13.                 LET S(p) be the set of test objectives covered by p
14.                 For each test case t in T where S(t)  S(p)
15.                     T  T − {t}
16.                 End
17.                 T T  {p}
18.                 MARK c as covered
19.                 BREAK
20.             End
21.         End
22.     End
23. End
24. RETURN T
```

شکل ۵: شبه کد الگوریتم تولید مورد آزمون مبتنی بر وارسی مدل.

۳ ۵ -تولید مورد آزمون تصادفی

آزمون تصادفی [۲۸]، یک روش مبتنی بر جستجو و از اساسی‌ترین و مشهورترین روش‌های آزمون است که به‌ویژه برای ارزیابی کارایی سایر راهکارهای آزمون به کار می‌رود. در این روش، تا زمانی که اهداف آزمون برآورده شود و یا بودجه آزمون تمام شود موارد آزمون کاملاً به‌صورت تصادفی تولید می‌شوند. آزمون تصادفی زمانی که راه‌حل در بخش کوچکی از فضای حالت قرار دارد نتایج بسیار ضعیفی دارد. افزون بر این، معمولاً منجر به تولید تعداد زیادی مورد آزمون با طول بالا می‌شود که اجرای آن‌ها مقرون به‌صرفه نیست.

۴ -راهکار پیشنهادی

برای بهبود بخشیدن به چالش‌های تولید آزمون مبتنی بر وارسی مدل، مسئله تولید مورد آزمون را به کمک الگوریتم جستجوی پرتو به یک

مسئله بهینه‌سازی تبدیل کرده‌ایم (شکل ۶). در راهکار پیشنهادی، توصیفات تبدیل گراف را به‌عنوان مدل سیستم تحت آزمون در نظر می‌گیریم. هر قانون موجود در این توصیفات می‌تواند معرف یک سرویس در سیستم‌های مبتنی بر معماری سرویس‌گرا، یک تابع در سیستم‌های شئ‌گرا و یا یک رخداد در سیستم‌های تعاملی یا ایمنی بحرانی باشد.

۴ ۱ -نحوه نمایش راه‌حل

در الگوریتم پرتو، هر راه‌حل مسئله (یک مورد آزمون) را به‌صورت یک آرایه از اعداد صحیح کدگذاری می‌کنیم. هر یک از این اعداد نمایانگر شماره گذر قانون خروجی از یک حالت در فضای حالت گراف است که می‌تواند از بازه (0,N) انتخاب شود. متغیر N حداکثر تعداد گذر قانون خروجی از یک حالت در کل مسئله است و از آنجا که این مقدار برای هر مسئله متفاوت است مقدار آن را به‌مرور زمان که فضای حالت کاوش می‌شود در صورت نیاز به‌روزرسانی می‌کنیم. قبل از نمایش موارد آزمون نهایی به کاربر به‌جای اعداد صحیح موجود در هر مورد آزمون برچسب فراخوانی قانون مربوط به آن گذر خروجی را قرار می‌دهیم (مانند جدول ۱). به‌علاوه اگر آن قانون پارامتر ورودی هم داشته باشد ابزار GROOVE این مقادیر را در اختیار ما قرار می‌دهد. این مقادیر یا از گراف میزبان مهیا می‌شوند و یا به‌واسطه اجرای قوانین به دست می‌آیند.

۴ ۲ -تابع شایستگی

در الگوریتم جستجوی پرتو شایستگی هر راه‌حل t به کمک تابع F در تعریف (۳) سنجیده می‌شود.

تعریف (۳): $F(t)$ = تعداد اهداف پوشش یافته توسط مورد آزمون

برای تشخیص اهداف پوشش یافته توسط یک مورد آزمون، مطابق شبه‌کد ارائه شده در شکل ۶ عمل می‌کنیم. برای هر مورد آزمون t، ابتدا فضای حالت را کاوش می‌کنیم EPath(t). هر دو زوج متوالی از فراخوانی قوانین موجود در این مسیر کاوش شده (d,u) را در نظر می‌گیریم. اگر زوج (d,u) جزء اهداف آزمون پوشش نیافته است (خط ۶)، بسته به اینکه معیار پوشش چه باشد حالات زیر بررسی می‌شود:

- معیار ۱: اگر بین عناصر ایجاد شده توسط گذر قانون d و عناصر خوانده شده توسط گذر قانون u اشتراکی وجود داشت هدف آزمون جاری پوشش یافته است (خط ۸).
- معیار ۲: اگر بین عناصر ایجاد شده توسط گذر قانون d و عناصر حذف شده توسط گذر قانون u اشتراکی وجود داشت هدف آزمون جاری پوشش یافته است (خط ۱۱).
- معیار ۳: اگر بین عناصر ایجاد شده توسط گذر قانون d و عناصر به‌روز شده توسط گذر قانون u اشتراکی وجود داشت هدف آزمون جاری پوشش یافته است (خط ۱۴).
- معیار ۴: ترکیب حالات ۱، ۲ و ۳ انجام می‌شود.

۴ ۳ -راهکار پیشنهادی برای تولید مورد آزمون

شکل ۷ طرح کلی راهکار پیشنهادی ما را نشان می‌دهد. به‌عنوان ورودی، توصیفات تبدیل گراف، معیار پوشش، حداکثر تعداد موارد آزمون، حداکثر طول موارد آزمون و نیز عرض پرتو از کاربر دریافت می‌شود. خروجی نیز مجموعه آزمونی با بیشترین پوشش و کمترین اندازه است. مراحل این راهکار بدین شرح است:

۱. در آغاز، با در نظر گرفتن مدل توصیفات تبدیل گراف، گراف جریان داده را به‌صورت ایستا به دست می‌آوریم و اهداف آزمون مربوط به معیار پوشش درخواستی را استخراج می‌کنیم (مطابق بخش ۳-۲).

۲. در ادامه، الگوریتم جستجوی پرتو ضمن مقداردهی اولیه به پارامتر خود، حالت شروع را به‌عنوان مسیر اولیه در نظر می‌گیرد و سپس همسایه‌های حالت فعلی را به مسیر فعلی اضافه می‌کند که حاصل آن مجموعه‌ای از مسیرها است. حالا، فهرستی از اهداف آزمونی که توسط هر یک از مسیرها قابل پوشش است محاسبه و ذخیره می‌شود (مطابق بخش ۴-۲). اندازه این فهرست، معرف میزان شایستگی آن مسیر (مورد آزمون) است (مطابق تعریف (۳)).

۳. در این مرحله، اگر طبق تعریف ۴ به پوشش ۱۰۰٪ دست یافته باشیم و یا حداکثر طول موارد آزمون حاصل شده باشد به گام ۵ می‌رویم. در غیر این صورت موارد آزمون بر اساس شایستگی، مرتب نزولی می‌شوند و به‌اندازه عرض پرتو بهترین مسیرها را ذخیره می‌کنیم و باقی آن‌ها را نادیده می‌گیریم.

تعریف (۴):

۴. الگوریتم پرتو، برای هر مسیر ذخیره شده t، همسایه‌های آخرین حالتش S_i را به مسیر فعلی اضافه می‌کند که حاصل آن مجموعه‌ای از مسیرها t_i است. فهرست اهداف پوشش یافته مسیر t به تمام مسیرهای t_i منتج از آن، اضافه می‌شود. سپس برای هر مسیر t_i صرفاً بررسی می‌شود که آیا حالت آخر آن یعنی S_i موجب پوشش اهداف آزمون جدیدی می‌شود یا خیر. در صورت یافتن چنین اهدافی، آن‌ها را به فهرست اهداف پوشش یافته مسیر t_i اضافه می‌کنیم. دستاورد این روش، کاهش چشمگیر میزان پردازش و نیاز به ذخیره‌سازی است. برای بررسی شرط خاتمه به مرحله ۳ برمی‌گردیم.

۵. مسیرهای ذخیره شده، مجموعه آزمون ما را تشکیل می‌دهند. در این مرحله با حفظ میزان پوشش، اگر مورد آزمونی وجود دارد که تمام اهداف پوشش یافته آن به‌وسیله یک مورد آزمون دیگر از این مجموعه قابل پوشش است، آن را حذف می‌کنیم. سپس از هر مورد آزمون، تنها یک دنباله فراخوانی با شروع از ابتدای آن تا جایی که حاوی اهداف پوشش باشد استخراج

می‌شود. در این مرحله، یک مجموعه آزمون با بیشترین پوشش، کمترین تعداد و کمترین طول مورد آزمون خواهیم داشت که این همان هدف ما از انجام پژوهش جاری است.

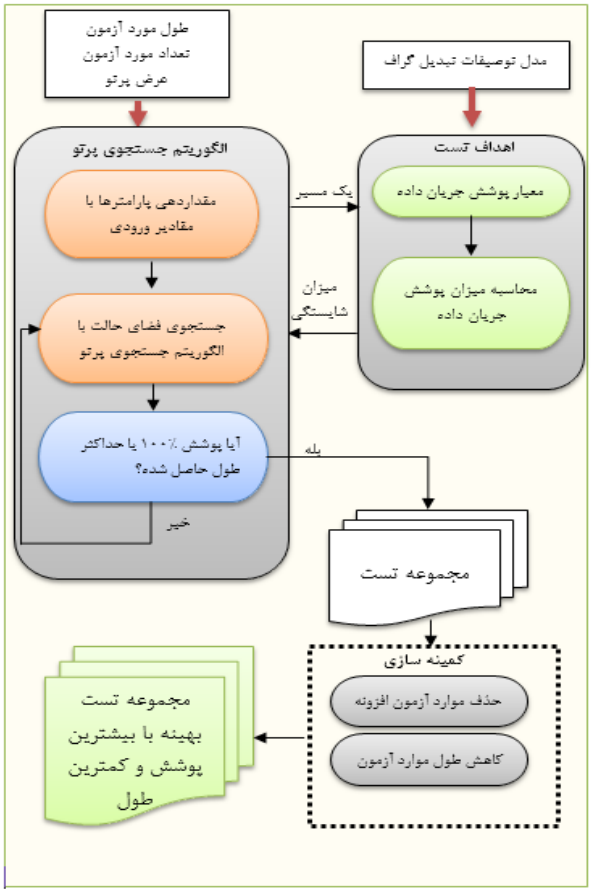
```
Input: Test suite T , Coverage criterion D;
Output: Fitness value F;

Let O be the test objectives, including all dependent pairs of rules with respect D;
Let C be the covered objectives. At first, it is empty;
For each test case t in T
    Let EPath(t) be the explored path by t;
    For each subsequent pair of rule transitions (d, u), namely p, in EPath(t)
        If p ∈ O and p ∉ C
            If D equals to Create-Read
                If d.addedHostEntities ∩ u.readHostEntities ≠ ∅ Then Add p to C ;

            Else If D equals to Create-Delete
                If d.addedHostEntities ∩ u.removedHostEntities ≠ ∅ Then Add p to C;

            Else If D equals to Create-Update
                If d.addedHostEntities ∩ u.updatedHostEntities ≠ ∅ Then Add p to C;
            End
        End
    End
End
F ← |C|;
Return F;
```

شکل ۶: شبه کد تابع شایستگی.



شکل ۷: طرح کلی راهکار پیشنهادی.

۵- ارزیابی

برای ارزیابی راهکار ارائه شده، آن را در قالب یک راهبرد جدید کاوش فضای حالت به ابزار GROOVE اضافه کردیم. این راهبرد به‌عنوان ورودی، توصیفات تبدیل گراف مسئله، حداکثر اندازه مجموعه آزمون و عرض پرتو را از کاربر دریافت کرده و در نهایت موارد آزمون بهینه را به همراه تحلیل‌های آماری چاپ می‌کند. به‌عنوان مطالعه موردی، توصیفات تبدیل گراف سه سیستم سرویس‌گرا که فضای حالت بزرگی دارند را انتخاب کردیم: سیستم ردیابی خطای نرم‌افزار [۲۹]، سیستم آژانس مسافرتی [۳۰] و سیستم فروشگاه آنلاین [۳۱]. نتایج حاصل از آزمایش‌ها را با سایر روش‌های تولید آزمون موجود مانند راهکار [۱۴]، تولید مورد آزمون تصادفی و راهکار وارسی مدل مقایسه خواهیم کرد. به دلیل ماهیت تخمینی الگوریتم‌های فرامکاشفه‌ای، هر آزمایش ۱۰ بار تکرار شده و میانگین نتایج را ثبت کرده‌ایم. علاوه براین، در تجزیه و تحلیل‌هایمان از آزمون‌های آماری Mann-Whitney و آزمون U استفاده می‌کنیم که در آن کارایی الگوریتم به کمک روش استانداردی به نام اندازه مؤثر[۲۱] $\hat{A}_{12}$ Vargha and Delaney تعیین می‌شود [۳۲]. $\hat{A}_{12}$ اندازه مؤثر غیرپارامتریک است که ما از آن برای مقایسه احتمال مقادیر بیشتر (یا کمتر) برای پوشش (یا اندازه مجموعه آزمون) دو روش استفاده می‌کنیم. به این صورت که اگر $\hat{A}_{12} < 0.5$ باشد رخداد نتایج روش اول احتمال کمتری نسبت به روش دوم دارد، مقدار $\hat{A}_{12} = 0.5$ یعنی احتمال برابر و $\hat{A}_{12} > 0.5$ نیز نشان‌دهنده احتمال بیشتر روش اول است. افزون بر این، اگر مشاهده شود که مقدار P-value (سطح معنی‌داری) کمتر از ۰/۰۵ است، به این معنی است که تفاوت قابل توجهی بین نتایج دو روش وجود دارد، در غیر این صورت نیاز به آزمایش‌های بیشتری برای قضاوت وجود دارد. تمام آزمایش‌ها، بر روی یک سیستم با مشخصات Intel(R) Core(TM) i5-2450M CPU @ 2.50 GHz, 4 GB RAM اجرا شده‌اند.

۵-۱- مطالعه موردی اول: سیستم ردیابی خطای نرم‌افزار

یک سیستم ردیابی خطای نرم‌افزار برای مدیریت بهتر تصحیح خطاهای موجود در یک نرم‌افزار استفاده می‌شود. مدیریت پروژه، کاربر و روابط بین آن‌ها از امکانات پایه این سیستم است. علاوه بر این، می‌توان به کاربران نقش‌های گوناگونی (مدیر اصلی، توسعه‌دهنده و

آزمون کننده) تخصیص داد. هر نقش، مجاز به انجام فعالیت‌های خاصی در سیستم خواهد بود. گروه توسعه‌دهنده و نیز آزمون‌کنندگان نرم‌افزار، به کمک این سیستم می‌توانند با یکدیگر تعامل مناسبی برای گزارش خطا، اولویت بندی و تصحیح آن داشته باشند.

جدول ۳، میانگین نتایج (درصد پوشش) روش‌های مختلف تولید مورد آزمون را بر روی مطالعه موردی سیستم ردیابی خطای نرم‌افزار نشان می‌دهد. همان‌طورکه ملاحظه می‌شود روش وارسی مدل حتی در گراف میزبان ساده نیز دچار انفجار فضای حالت می‌شود. در مقابل راهکار پیشنهادی در تمام موارد نسبت به روش تصادفی به پوشش بالاتری دست یافته است و به دلیل هدایت هوشمند تولید فضای حالت حتی در گراف میزبان پیچیده نیز با کمبود حافظه مواجه نشده است.

ستون هفتم، هشتم و نهم از جدول ۳، مقادیر اندازه مؤثر میانگین پوشش روش پیشنهادی را در مقایسه با الگوریتم خفاش، الگوریتم جستجوی نیروی گرانشی و روش تصادفی نشان می‌دهند. در مواردی که اختلاف معناداری وجود داشته (مقدار P-value کمتر از 0.05) مقادیر پررنگ نشان داده شده‌اند. همان‌گونه که مشاهده می‌شود در مقایسه با آزمون تصادفی از ۴ مورد، روش پیشنهادی ما در ۳ مورد عملکرد بهتری داشته است و تنها در ۱ مورد نیاز به آزمایش‌های بیشتری برای قضاوت است. در مقایسه با آزمون مبتنی بر جستجو نیز در ۲ مورد روش ما عملکرد بهتری دارد و در ۲ مورد آزمون مبتنی بر جستجو عملکرد بهتری دارد. جدول ۴ میانگین نتایج (تعداد مورد آزمون، طول مورد آزمون) روش‌های مختلف تولید مورد آزمون را بر روی مطالعه موردی سیستم ردیابی خطای نرم‌افزار نشان می‌دهد؛ زمانی که تفاوت آماری معنی‌داری در اندازه پوشش در روش‌های مختلف وجود ندارد می‌توان آن‌ها را از نظر اندازه مجموعه آزمون با یکدیگر مقایسه کرد. آزمایشات را روی سه گراف میزبان مختلف با ۹، ۱۴ و ۲۱ شئ اجرا کردیم. شکل ۸ میزان مصرف بودجه آزمون توسط روش پیشنهادی را بر روی گراف‌های شروع مختلف از سیستم ردیابی خطای نرم‌افزار روی معیار ۴ نشان می‌دهد که ملاحظه می‌شود زمان کمی صرف تولید داده آزمون شده است. علاوه بر این، شکل ۹ به مقایسه میانگین میزان پوشش اهداف آزمون حاصل از روش تولید مورد آزمون تصادفی و روش پیشنهادی روی گراف‌های شروع مختلف می‌پردازد که برتری روش پیشنهادی مشهود است؛ روش پیشنهادی در تمام گراف‌ها اعم از ساده و پیچیده پوشش ثابت دارد.

۵ ۴ مطالعه موردی دوم- سیستم آژانس مسافرتی

در این سیستم، آژانس‌های مسافرتی متناسب با نیاز مسافر، خدماتی از قبیل رزرو بلیط هواپیما و رزرو هتل را ارائه می‌دهند.

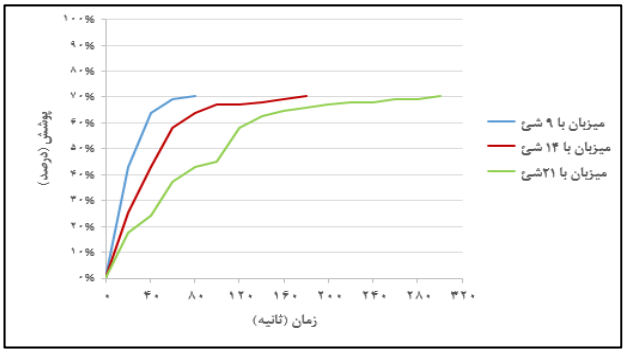
میانگین نتایج (درصد پوشش) از اجرای روش‌های مختلف تولید مورد آزمون روی مطالعه موردی آژانس مسافرتی در جدول ۵ قابل مشاهده است. همان‌طورکه ملاحظه می‌شود روش وارسی مدل در این سیستم نیز دچار انفجار فضای حالت می‌شود. ستون هفتم، هشتم و نهم از جدول ۵، مقادیر اندازه مؤثر میانگین پوشش روش پیشنهادی را در مقایسه با الگوریتم خفاش، الگوریتم جستجوی نیروی گرانشی و روش تصادفی نشان می‌دهند.

جدول ۳: مقایسه میانگین پوشش روش‌های مختلف تولید مورد آزمون بر روی سیستم ردیابی خطای نرم‌افزار.

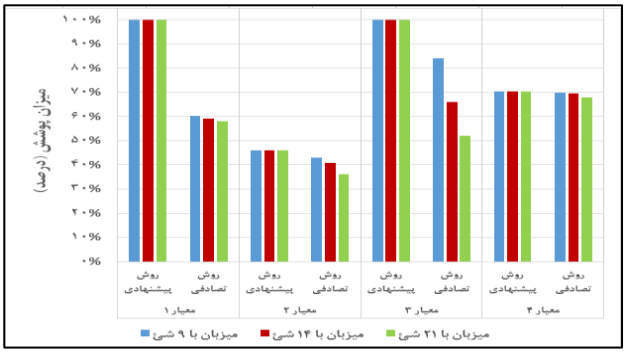
ضریب تأثیر			آزمون مبتنی بر وارسی مدل	آزمون تصادفی	آزمون مبتنی بر جستجو [۱۴]		روش پیشنهادی	
روش پیشنهادی و آزمون تصادفی	روش پیشنهادی و الگوریتم جستجوی نیروی گرانشی	روش پیشنهادی و الگوریتم خفاش			الگوریتم جستجوی نیروی گرانشی	الگوریتم خفاش		
				پوشش (درصد)	پوشش (درصد)	پوشش (درصد)	پوشش (درصد)	معیار
۱	۱	۱	کمبود حافظه	۶۰/۲۷	۸۶/۷۱	۶۵/۷۵	۱۰۰	۱
۰/۶	۰	۰		۴۳/۰۷	۶۹/۲۲	۶۵/۳۷	۴۶/۱۵	۲
۰/۷۵	**۰/۹۵**	**۰/۹**		۸۴	۷۸	۸۰	۱۰۰	۳
۰/۸	**۰/۱**	**۰/۰۵**		۶۹/۹۱	۸۶/۱۴	۷۵/۸۱	۷۰/۳۲	۴

جدول ۴: مقایسه میانگین اندازه مجموعه آزمون روش‌های مختلف تولید مورد آزمون بر روی سیستم ردیابی خطای نرم‌افزار .

آزمون مبتنی بر وارسی مدل	آزمون تصادفی		آزمون مبتنی بر جستجو [۱۴]				روش پیشنهادی		
			الگوریتم جستجوی نیروی گرانشی		الگوریتم خفاش				
	اندازه موارد آزمون	تعداد مورد آزمون	اندازه موارد آزمون	تعداد مورد آزمون	اندازه موارد آزمون	تعداد مورد آزمون	اندازه موارد آزمون	تعداد مورد آزمون	معیار
کمبود حافظه	۳۸/۱۹	۱۱/۱	۴۷/۵۸	۳/۴	۴۷/۶	۳/۳	۳۸	۹	**۱**
	۱۳/۸۹	۴/۴	۴۴/۶۴	۲/۸	۴۲/۹۶	۲/۸	۱۳	۱	**۲**
	۱۱/۶۷	۴	۱۱/۱	۱/۹	۱۱/۹	۲	۱۰	۱	**۳**
	۴۱/۳	۱۴/۲	۴۹/۳۷	۴/۳	۴۹/۵	۳/۴	۵۰	۱	**۴**



شکل ۸: مقایسه میزان مصرف بودجه آزمون توسط روش پیشنهادی بر روی گراف‌های شروع مختلف سیستم ردیابی خطای نرم‌افزار.



شکل ۹: مقایسه آزمون تصادفی و روش پیشنهادی از نظر میزان پوشش با افزایش پیچیدگی گراف شروع در سیستم ردیابی خطای نرم‌افزار.

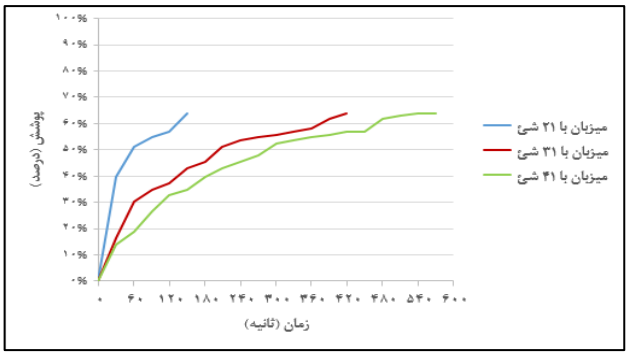
در مواردی که اختلاف معناداری وجود داشته مقادیر پررنگ نشان داده شده‌اند. همان‌گونه که مشاهده می‌شود در مقایسه با روش تصادفی از ۴ مورد، روش پیشنهادی ما در ۲ مورد و روش تصادفی در ۲ مورد عملکرد بهتری داشته است. در مقایسه با آزمون مبتنی بر جستجو نیز در ۱ مورد روش ما و در ۳ مورد روش آزمون مبتنی بر جستجو عملکرد بهتری دارد. جدول ۶ میانگین نتایج (تعداد مورد آزمون، طول مورد آزمون) روش‌های مختلف تولید مورد آزمون را بر روی مطالعه موردی سیستم آژانس مسافرتی نشان می‌دهد. آزمایشات را روی سه گراف میزبان مختلف شامل ۲۱، ۳۱ و ۴۱ شئ اجرا کردیم، شکل ۱۰ میزان مصرف بودجه آزمون توسط روش پیشنهادی را بر روی گراف‌های شروع مختلف از سیستم آژانس مسافرتی روی معیار ۴ نشان می‌دهد که ملاحظه می‌شود زمان کمی صرف تولید داده آزمون شده است. علاوه بر این، شکل ۱۱ به مقایسه میانگین میزان پوشش اهداف آزمون حاصل از روش تولید مورد آزمون تصادفی و روش پیشنهادی روی گراف‌های شروع مختلف می‌پردازد.

جدول ۵: مقایسه میانگین پوشش روش‌های مختلف تولید مورد آزمون بر روی سیستم آژانس مسافرتی.

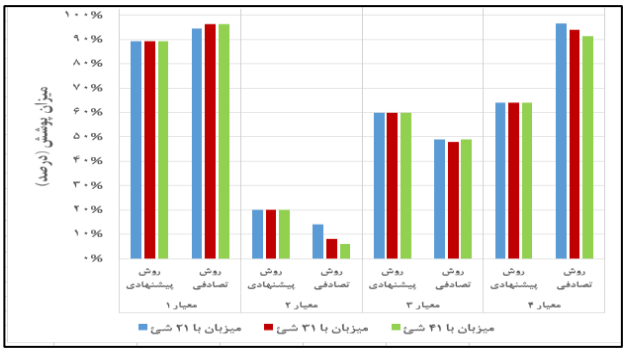
ضریب تأثیر			آزمون مبتنی بر وارسی مدل	آزمون تصادفی	آزمون مبتنی بر جستجو [۱۴]		روش پیشنهادی	
روش پیشنهادی و آزمون تصادفی	روش پیشنهادی و الگوریتم جستجوی نیروی گرانشی	روش پیشنهادی و الگوریتم خفاش			الگوریتم جستجوی نیروی گرانشی	الگوریتم خفاش		
				پوشش (درصد)	پوشش (درصد)	پوشش (درصد)	پوشش (درصد)	معیار
۰/۰۵	۰/۸۵	۱	کمبود حافظه	۹۴/۵۳	۸۳/۶۳	۶۸/۴۸	۸۹/۳۹	۱
۰/۸	۰	۰		۱۴	۷۵/۵۴	۶۵	۲۰	۲
۰/۸۵	۰/۴	۰/۴۵		۴۹	۶۲	۶۰	۶۰	۳
۰	۰	۰/۱		۹۶/۵	۸۸/۸۳	۷۰/۴۵	۶۳/۹۵	۴

جدول ۶: مقایسه میانگین اندازه مجموعه آزمون روش‌های مختلف تولید مورد آزمون بر روی سیستم آژانس مسافرتی.

آزمون مبتنی بر وارسی مدل	آزمون تصادفی		آزمون مبتنی بر جستجو [۱۴]				روش پیشنهادی	
			الگوریتم جستجوی نیروی گرانشی		الگوریتم خفاش			
	اندازه موارد آزمون	تعداد مورد آزمون	اندازه موارد آزمون	تعداد مورد آزمون	اندازه موارد آزمون	تعداد مورد آزمون	اندازه موارد آزمون	تعداد مورد آزمون	معیار
کمبود حافظه	۴۲/۵۴	۱۲/۷	۳۸/۳۷	۲/۹	۳۷/۹۶	۲/۷	۴۲	۱	۱
	۷/۴۳	۱/۴	۳۴/۴۵	۲	۳۱/۴	۱/۸	۴	۲	۲
	۱۰/۲۶	۴/۶	۳۷/۴	۱	۳۶/۸	۱	۱۲	۱	۳
	۳۴/۶۹	۱۶/۲	۳۸/۳۷	۲/۹	۳۸/۲۷	۳/۷	۳۹	۱	۴



شکل ۱۰: مقایسه میزان مصرف بودجه آزمون توسط روش پیشنهادی بر روی گراف‌های شروع مختلف سیستم آژانس مسافرتی.



شکل ۱۱: مقایسه آزمون تصادفی و روش پیشنهادی از نظر میزان پوشش با افزایش پیچیدگی گراف شروع در سیستم آژانس مسافرتی.

۵ ۳ -مطالعه موردی سوم- سیستم فروشگاه آنلاین

در سیستم فروشگاه آنلاین خرید محصولات از طریق اینترنت برای مشتریان فراهم می‌شود. مشتریان می‌توانند فهرست محصولات را مشاهده و با داشتن کارت اعتباری محصولات مورد نیاز را سفارش دهند. پرداخت با استفاده از یک کارت اعتباری انجام می‌شود. پس از پرداخت، سفارشات به آدرس مشتری فرستاده می‌شود. مشتریان تنها می‌توانند سفارشات خود را قبل از ارسال سفارشات لغو کنند.

جدول ۷، میانگین نتایج (درصد پوشش) روش‌های مختلف تولید مورد آزمون را بر روی این سیستم نشان می‌دهد. همان‌طورکه ملاحظه می‌شود روش وارسی مدل در این سیستم نیز دچار انفجار فضای حالت می‌شود و حتی در گراف میزبان ساده نیز موفق به تولید مورد آزمون نمی‌شود. در مقابل راهکار پیشنهادی در بیشتر موارد نسبت به روش تصادفی به پوشش بالاتری دست یافته است. ستون هفتم، هشتم و نهم از جدول ۷، مقادیر اندازه مؤثر میانگین پوشش روش پیشنهادی را در مقایسه الگوریتم خفاش، الگوریتم جستجوی نیروی گرانشی و روش تصادفی نشان می‌دهند. در مواردی که اختلاف معناداری وجود داشته مقادیر پررنگ نشان داده شده‌اند. همان‌گونه که مشاهده می‌شود در مقایسه با روش تصادفی از ۴ مورد، روش پیشنهادی ما در ۳ مورد عملکرد بهتری داشته است و در ۱ مورد عملکرد برابر دارند. در مقایسه با آزمون مبتنی بر جستجو نیز در ۳ مورد روش ما و در ۱ مورد آزمون مبتنی بر جستجو عملکرد بهتری دارد. جدول ۸ میانگین نتایج (تعداد مورد آزمون، طول مورد آزمون) روش‌های مختلف تولید مورد آزمون را بر روی مطالعه موردی سیستم فروشگاه آنلاین نشان می‌دهد. در ۱ موردی که روش پیشنهادی ما و روش تصادفی عملکرد برابر داشتند مشاهده می‌شود که روش پیشنهادی اندازه مجموعه آزمون کمتری دارد یعنی عملکرد بهتری دارد. آزمایشات را روی سه گراف میزبان مختلف با ۱۳، ۲۳ و ۳۳ شئ اجرا کردیم. شکل ۱۲ میزان مصرف بودجه آزمون توسط روش پیشنهادی را بر روی گراف‌های شروع مختلف از سیستم فروشگاه آنلاین روی معیار ۴ نشان می‌دهد که ملاحظه می‌شود زمان کمی صرف تولید داده آزمون کرده است. علاوه بر این، شکل ۱۳ به مقایسه میانگین میزان پوشش اهداف آزمون حاصل از روش تولید مورد آزمون تصادفی و روش پیشنهادی روی گراف‌های شروع مختلف می‌پردازد که برتری روش پیشنهادی مشهود است؛ روش پیشنهادی در تمام گراف‌ها اعم از ساده و پیچیده پوشش ثابت دارد.

۶ -نتیجه‌گیری و کارهای آتی

هدف از این پژوهش، ارائه راهکاری بهینه برای بهبود بخشیدن به چالش‌های راهکار تولید مورد آزمون به کمک ابزار وارسی مدل است. انفجار فضای حالت یکی از مهم‌ترین چالش‌هایی است که این راهکار با آن روبرو است. روش پیشنهادی ما که از رویکرد آزمون جعبه سیاه، دنباله آزمون را استخراج می‌کند با به‌کارگیری الگوریتم جستجوی پرتو نه‌تنها قادر به مقابله با مسئله انفجار فضای حالت است بلکه برخلاف سایر الگوریتم‌ها با ساختار نه‌چندان پیچیده خود، از نظر میزان پوشش و اندازه، مجموعه آزمون‌هایی به‌مراتب بهتر تولید می‌کند.

در واقع الگوریتم جستجوی پرتو به دلیل اینکه تمام فضای حالت را باز نمی‌کند و در هر مرحله از فضای حالت جستجو شده بهترین مسیرها را نگهداری می‌کند (مسیرهایی با میزان پوشش بالاتر)، این امر باعث می‌شود که فضای جستجو به‌طور هوشمند هدایت شود و میزان پوشش مسیر حتی در گراف‌های پیچیده نیز تغییر نکند.

افزون بر این، از آنجا که در جستجوی پرتو تعداد حالت‌ها به‌صورت خطی با عمق جستجو افزایش می‌یابد، این الگوریتم اغلب عملکرد بهتری در مدل‌های بزرگ و یا در مدل‌هایی با مسیر محاسباتی عمیق، دارد. به‌منظور ارزیابی کارایی، روش پیشنهادی در ابزار وارسی مدل توصیفات تبدیل گراف GROOVE پیاده‌سازی شد و روی سه مطالعه موردی در حوزه سیستم‌های سرویس‌گرا مورد آزمایش قرار گرفت. نتایج آزمایش‌ها حاکی از آن است که راهکار ما نسبت به راهکارهای موجود، موفق به تولید مجموعه آزمون‌هایی با میزان پوشش بالاتر و در زمان کمتر می‌شود.

نتایج آزمایش‌ها حاکی از آن است که راهکار ما نسبت به راهکارهای موجود، موفق به تولید مجموعه آزمون‌هایی با میزان پوشش بالاتر و در زمان کمتر می‌شود.

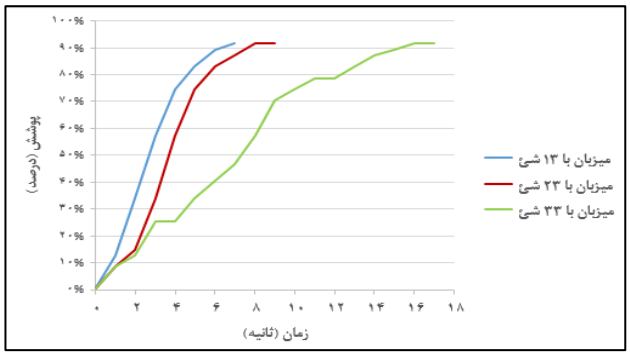
عموماً الگوریتم‌های ترکیبی نتایج بهتری در حل مسائل بهینه‌سازی ارائه می‌دهند. در آینده قصد داریم با ترکیب الگوریتم جستجوی پرتو و سایر الگوریتم‌های فرامکاشفه‌ای راهکار پیشنهادی را تقویت بخشیم. همچنین به کمک استخراج خودکار کد پیاده‌سازی از توصیفات تبدیل گراف مسئله و سپس اجرای موارد آزمون می‌توان پاسخ سیستم را با پاسخ مورد انتظار مقایسه و نتیجه آزمون را به کاربر اعلام کرد.

جدول ۷: مقایسه میانگین پوشش روش‌های مختلف تولید مورد آزمون بر روی سیستم فروشگاه آنلاین.

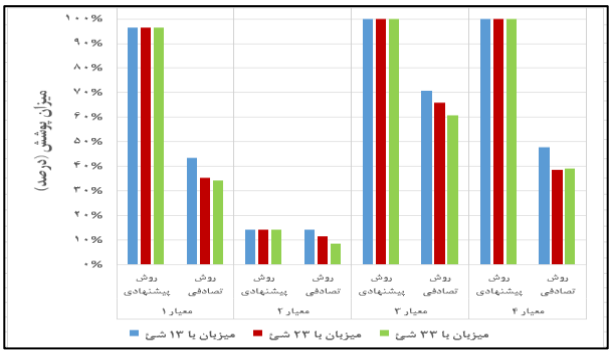
ضریب تأثیر			آزمون مبتنی بر وارسی مدل	آزمون تصادفی	آزمون مبتنی بر جستجو [۱۴]		روش پیشنهادی	معیار
روش پیشنهادی و آزمون تصادفی	روش پیشنهادی و الگوریتم جستجوی نیروی گرانشی	روش پیشنهادی و الگوریتم خفاش		پوشش (درصد)	الگوریتم جستجوی نیروی گرانشی	الگوریتم خفاش	پوشش (درصد)	
۱	۱	۱	کمبود حافظه	۴۳/۴۱	۸۳/۵۶	۹۰/۳۵	۹۶/۴۲	۱
۰/۵	۰	۰		۱۴/۲۸	۵۵/۷۱	۶۵/۷	۱۴/۲۸	۲
۱	۰/۹۵	۱		۷۰/۸۳	۷۹/۱۶	۷۴/۱۶	۱۰۰	۳
۱	۰/۹۵	۰/۹۵		۴۷/۸۹	۷۹/۵۶	۸۲/۳۳	۹۱/۴۸	۴

جدول ۸: مقایسه میانگین اندازه مجموعه آزمون روش‌های مختلف تولید مورد آزمون بر روی سیستم فروشگاه آنلاین.

آزمون مبتنی بر وارسی مدل	آزمون تصادفی		آزمون مبتنی بر جستجو [۱۴]				روش پیشنهادی		معیار
			الگوریتم جستجوی نیروی گرانشی		الگوریتم خفاش				
	اندازه موارد آزمون	تعداد مورد آزمون	اندازه موارد آزمون	تعداد مورد آزمون	اندازه موارد آزمون	تعداد مورد آزمون	اندازه موارد آزمون	تعداد مورد آزمون	
کمبود حافظه	۱۶/۳۷	۴/۲	۱۹/۶۳	۱/۹	۱۹/۸۷	۲/۴	۱۸	۱	۱
	۸/۰۸	۱	۱۷/۰۸	۱/۲	۱۶/۱۲	۱/۶	۲	۱	۲
	۱۶/۳۶	۴/۷	۱۹/۲	۱/۵	۱۸/۹۴	۱/۷	۱۷	۱	۳
	۱۷/۸۳	۵/۱	۱۹/۹۵	۲	۱۹/۶۲	۲/۴	۱۹	۱	۴



شکل ۱۲: مقایسه میزان مصرف بودجه آزمون توسط روش پیشنهادی بر روی گراف‌های شروع مختلف سیستم فروشگاه آنلاین.



شکل ۱۳: مقایسه آزمون تصادفی و روش پیشنهادی از نظر میزان پوشش با افزایش پیچیدگی گراف شروع در سیستم فروشگاه آنلاین.

مراجع

[1] P. Ammann and J. Offutt, *Introduction to software testing*, 1 ed. United States of America, New York, Cambridge University Press, April 2008.

[2] L. Zhao and W. Luo, "An algorithm for reducing test suite based on interface parameters," International Conference on Computational Intelligence and Software Engineering, pp. 1-4, Wuhan, China, December 2010.

[۳] سجاد اسفندیاری و وحید رافع، «راهکاری نوین جهت تولید دنباله آزمون کمینه در فرآیند آزمون نرم‌افزار با ترکیب الگوریتم‌های جستجوی تپه‌نوردی و جستجوی خفاش»، مجله مهندسی برق دانشگاه تبریز، جلد ۴۶، شماره ۳ صفحه ۲۵-۳۵، پاییز ۱۳۹۵.

[4] M. Utting, B. Legeard, F. Bouquet, E. Fourneret, F. Peureux, and A. Vernotte, "Recent advances in model-based testing," Advances in Computers. vol. 101, pp. 53-120, Elsevier, January 2016.

[5] C. Baier and J.-P. Katoen, *Principles of model checking*, Massachusetts,MIT Press, January 2008.

[6] M. P. E. Heimdahl, S. Rayadurgam, W. Visser, G. Devaraj, and J. Gao, "Auto-generating test sequences using model checkers: a case study," Third International Workshop on Formal Approaches to Testing of Software, pp. 42-59 Berlin, Heidelberg: Springer, 2004.

[7] V. Rafe, M. Rahmani, and K. Rashidi, "A Survey on coping with the state space explosion problem in model checking," *International Research Journal of Applied and Basic Sciences*, vol. 4, no.3, pp. 1379-1384, 2013.

[8] G. Fraser, F. Wotawa, and P. Ammann, "Issues in using model checkers for test case generation," *Journal of Systems and Software*, vol. 82, no.9, pp. 1403-1418, September 2009.

[9] V. Rafe, M. Moradi, R. Yousefian, and A. Nikanjam, "A Meta-heuristic solution for automated refutation of complex software systems specified through graph transformations," Applied Soft Computing, vol. 33, no. c, pp. 136-149, August 2015.

[10] R. Yousefian, S. Aboutorabi, and V. Rafe, "A Greedy algorithm versus metaheuristic solutions to deadlock detection in graph transformation systems," Journal of Intelligent and Fuzzy Systems, vol. 31, no. 1, pp. 137-149, June 2016.

[11] X.-S. Yang, "Metaheuristic optimization: algorithm analysis and open problems," Proceedings of the 10th International Conference on Experimental Algorithms, vol. 6630 pp. 21-32, Crete, Greece, May 2011.

[12] D. Furcy and S. Koenig, "Limited discrepancy beam search," In Proceedings of the 19th International Joint Conference on Artificial Intelligence, pp. 125-131, Edinburgh, Scotland, August 2005.

[13] A. Rensink, "The GROOVE simulator: a tool for state space generation," Applications of Graph Transformations with Industrial Relevance, vol. 3062 pp. 479-485 Berlin, Heidelberg, Springer, 2004.

[14] M. A. Bokhari, T. Bormer, and M. Wagner, "An Improved beam-search for the test case generation for formal verification systems," Proceedings of the 7th International Symposium on Search Based Software Engineering, vol. 9275 pp. 77-92, Cham, 2015.

[15] B. Jiang and W. K. Chan, "Input-based adaptive randomized test case prioritization: A local beam search approach," Journal of Systems and Software, vol. 105, no. c, pp. 91-106, July 2015.

[۱۶] اکرم کلائی، تولید بهینه مورد آزمون مبتنی بر مدل برای توصیفات تبدیل گراف با استفاده از الگوریتم‌های فرامکاشفه‌ای، درجه عالی، دانشگاه اراک، اراک، صفحه ۱-۱۲۰، زمستان ۱۳۹۴.

[17] R. Heckel, T. Khan, and R. Machado, "Towards test coverage criteria for visual contracts," Proceedings of the Tenth International Workshop on Graph Transformation and Visual Modeling Techniques GTVMT - Electronic Communications of the EASST, vol. 41, pp. 1-15 Berlin, January 2011.

[18] C. Nebut, F. Fleurey, Y. L. Traon, and J.-M. Jezequel, "Automatic test generation: A use case driven approach," IEEE Transactions on Software Engineering, vol. 32, no. 3, pp. 140-155, March 2006.

[19] M. A. Saadtjoo and S. M. Babamir, "Optimizing cost Function in Imperialist competition algorithm for path coverage problem in software testing," *Journal of AI and Data Mining*, vol. 5, September 2017.

[20] S. Z. Waheed and U. Qamar, "Data flow based test case generation algorithm for object oriented integration testing," 6th IEEE International Conference on Software Engineering and Service Science (ICSESS), pp. 423–427, Beijing, China, September 2015.

[21] N. Nayak and D. P. Mohapatra, "Automatic test data generation for data flow testing using particle swarm optimization," Communications in Computer and Information Science, vol. 95, pp. 1-12, Berlin, Heidelberg, January 2010.

[22] R. Grzegorz, *Handbook of graph grammars and computing by graph transformation*, vol. 1, World Scientific, 1997.

[۲۳] مریم مردادی، رزا یوسفیان و وحید رافع، « ارائه راهکاری جهت مقابله با مشکل انفجار فضای حالت در سیستم‌های تبدیل گراف با استفاده از الگوریتم‌های پرندگان و جستجوی گرانشی»، مجله مهندسی برق دانشگاه تبریز، جلد ۴۵، شماره ۴ صفحه ۱۶۳-۱۷۷، زمستان ۱۳۹۴.

[24] R. Heckel, "Graph transformation in a nutshell," *Electronic Notes in Theoretical Computer Science*, vol. 148, no.1, pp. 187-198, February 2006.

[25] P. M. Herman, "A Data flow analysis approach to program testing," The Australian Computer Journal, vol. 8, no. 3, pp. 92-96, November 1976.

[26] A. Groce and W. Visser, "Heuristics for model checking java programs," *International Journal on Software Tools for Technology Transfer (STTT)*, vol. 6, no. 4, pp. 260-276, August 2004.

[27] T. Su, Z. Fu, G. Pu, J. He, and Z. Su, "Combining symbolic execution and model checking for data flow testing," Proceedings of the 37th International Conference on Software Engineering, vol. 1, pp. 654-665, Florence, Italy, May 2015.

[28] P. McMinn, "Search-based software testing: past, present and future," Proceedings of the 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops, pp. 153-163, April 2011.

[29] O. Runge, T. A. Khan, and R. Heckel, "Test Case Generation Using Visual Contracts," *Proceedings of the 12th International Workshop on Graph Transformation*

and Visual Modeling Techniques GTVMT- Electronic Communications of the EASST, vol. 58, Italy, 2013.

[30] V. Rafe, "Scenario-driven analysis of systems specified through graph transformations," Journal of Visual Languages & Computing, vol. 24, no. 2, pp. 136-145, April 2013.

[31] G. Engels, B. Güldali , and M. Lohmann, "Towards model-driven unit testing," Proceedings of the 2006 International Conference on Models in Software Engineering, vol. 4364, pp. 182-192, Italy, October 2006.

[32] A. Arcuri and L. Briand, "A practical guide for using statistical tests to assess randomized algorithms in software engineering," Proceedings of the 33rd International Conference on Software Engineering, pp. 1-10, Waikiki, Honolulu, HI, USA, May 2011.

زیرنویس‌ها

[1] Model Base Testing
[2] Coverage Criteria
[3] Model Checking
[4] State Space Explosion
[5] Beam Search Algorithm
[6] Graph Object Oriented Verification
[7] Gravitational Search Algorithm
[8] Unified Modeling Language
[9] Attributed Graph Grammar
[10] Planning Domain Definition Language
[11] Graph Transition System
[12] Type Graph
[13] Host Graph
[14] Rule
[15] Left Hand Side
[16] Right Hand Side
[17] Negative Application Condition
[18] Definition
[19] Breadth-first Search
[20] Linear Temporal Logic
[21] Effect Size