Semnan University

Journal of Modeling in Engineering

Journal homepage: https://modelling.semnan.ac.ir/

ISSN: 2783-2538

Research Article

Provide a Solution Based on Teacher and Student Learning Algorithm to Reduce Regression Test Cases

Mahmood Deypir [a,*], Amirhossein Mohammadpour [b]

[a] *Associate Professor, Faculty of Computer Engineering, Shahid Sattari Aeronautical University of Science and Technology, Tehran, Iran*

[b] *MSc, Department of Computer Engineering, South Tehran Branch, Faculty of Technical and Engineering, Islamic Azad University, Tehran, Iran*

PAPER INFO

Paper history:

Received: 05 February 2024
Revised: 08 March 2024
Accepted: 16 March 2024

Keywords:

Software test,
Optimization,
Regression test,
Teacher and student Learning algorithm.

ABSTRACT

The aim of selecting test items is to choose a subset that has the potential to detect errors due to changes within the software. In other words, the purposes of test selection methods is to reduce the number of test cases after changing the code and focus on identifying the modified parts of the program. Intelligent methods such as regression improve the accuracy of tests in software projects, and the use of optimization algorithms to find the optimal amount of test cases can be useful in terms of time and speed, and according to research by examining and optimizing this algorithm in the system. In this paper, a technique for reducing regression test cases based on teacher-student optimization method was presented. This method was studied in two stages of teacher (education phase) and student (learning phase) on the test set and was implemented with different parameters. The experimental results showed that the use of the teacher-student algorithm reduces the time required for the reduction parameters of the regression test to some extent, although it does not give us a definite answer and will give a near-optimal answer. Also, the results of teacher-student algorithm were compared with previous approaches of regression test case reduction. Experimental results show better average execution time for test case selection.

DOI: https://doi.org/10.22075/jme.2024.31194.2490

*** Corresponding author.**
E-mail address: mdeypir@ssau.ac.ir

How to cite this article:
Mousavi, A., & Kazemian, A. H. (2024). Vibration Analysis of Sandwich Panel Structures: A Homogenized Simulation Approach with Modal Analysis. Journal of Modeling in Engineering, 22(78), 17-30. doi: 10.22075/jme.2024.33202.2616

مقاله پژوهشی

ارائه راهکاری مبتنی بر الگوریتم یادگیری معلم و دانش آموز به منظور کاهش موارد آزمون رگرسیون

محمود دی پیر[۱،*] ، امیرحسین محمدپور[۲]

اطلاعات مقاله

دریافت مقاله: ۱۴۰۲/۱۱/۱۶
بازنگری مقاله: ۱۴۰۲/۱۲/۱۸
پذیرش مقاله: ۱۴۰۲/۱۲/۲۶

واژگان کلیدی:
آزمون نرم افزار،
بهینه سازی،
آزمون رگرسیون،
الگوریتم یادگیری معلم- دانش آموز.

چکیده

هدف انتخاب موارد آزمون این است که بتوان زیر مجموعه‌ای انتخاب شود که قابلیت بالقوه شناسایی خطاهای ناشی از تغییرات را داشته باشد. به عبارتی هدف روش‌های انتخاب موارد آزمون، کاهش تعداد موارد آزمون بعد از تغییر کد است و بر روی شناسایی بخش‌های اصلاح شده برنامه تمرکز دارد. روش‌های هوشمند مانند رگرسیون، دقت آزمون را در پروژه‌های نرم افزاری بهبود می‌بخشند و استفاده از الگوریتم‌های بهینه سازی در یافتن مقدار بهینه موارد آزمون می‌تواند از نظر زمان و سرعت هم مفید واقع شود. در این مقاله تکنیکی برای کاهش موارد آزمون رگرسیون مبتنی برروش بهینه سازی معلم- دانش آموز ارائه می‌شود. این روش از دو مرحله معلم(فاز آموزش) و دانش آموز (فاز یادگیری) روی مجموعه آزمون تشکیل شده و بر اساس پارامترهای مختلف پیاده‌سازی گردیده است. نتایج آزمایش‌ها نشان داد که استفاده از الگوریتم معلم- دانش آموز، زمان لازم برای کاهش موارد آزمون رگرسیون را تا حدی بهبود می‌بخشد، هر چند که جواب قطعی را به ما نمی‌دهد و جوابی نزدیک به بهینه را خواهد داد. نتایج حاصل از اجرای رویکرد پیشنهادی با روش‌های قبلی انتخاب موارد آزمون مقایسه شده و مشاهده شد که میانگین زمان اجرای موارد آزمون انتخابی، توسط آن بهتر است.

DOI: https://doi.org/10.22075/jme.2024.31194.2490

۱-مقدمه

تقریباً در تمام طول چرخه‌ی توسعه‌ی نرم‌افزار از فاز نیازمندی[۳] تا فاز نگهداری[۴]، آزمون نرم افزار یک فرآیند اصلی و پرهزینه می‌باشد که توسعه‌دهندگان باید به آن توجه ویژه داشته باشند. آزمون نرم‌افزار اینگونه تعریف می‌گردد:"فرآیند اجرای برنامه با هدف یافتن خطاهای آن". آزمون نرم‌افزار یک جزء مهم از فرآیند کلی‌تر تأیید کیفیت نرم‌افزار است و سازمان‌های بسیاری تا ۴۰٪ از منابع توسعه‌ی خود را صرف این فرآیند می‌کنند. در فرآیند توسعه‌ی نرم افزار، اصلاح و به‌روز رسانی نرم‌افزار امری اجتناب‌ناپذیر است. این تغییرات می‌تواند به دلایل مختلف ازجمله، اضافه کردن ویژگی جدید در برنامه، اصلاح بخشی از برنامه به منظور عملکرد بهتر سیستم، انتقال برنامه بر روی محیط جدید و غیره صورت گیرد. انجام تغییرات در کد برنامه می‌تواند باعث ایجاد اختلال عملکرد در سایر قسمت‌های تغییر نیافته‌ی برنامه و به‌طور کلی عدم صحت کارکرد کلی برنامه شود. بنابراین آزمون رگرسیون ضروریست تا صحت عملکرد برنامه بعد از هر اصلاح در

* پست الکترونیک نویسنده مسئول: mdeypir@ssau.ac.ir
۱. دانشیار، دانشکده رایانه، دانشگاه علوم و فنون هوایی شهید ستاری، تهران، ایران.
۲.کارشناسی ارشد، گروه مهندسی کامپیوتر، واحد تهران جنوب، دانشگاه آزاد اسلامی، تهران، ایران.

[3] requirements
[4] maintenance

استناد به این مقاله:
موسوی، امیر، و کاظمیان، امیرحسین. (۱۴۰۳). آنالیز ارتعاش ساندویچ پنل با رویکرد شبیه‌سازی همگن با تجزیه و تحلیل مودال. مدل سازی در مهندسی، ۲۲(۷۸)، ۱۷-۳۰. doi: 10.22075/jme.2024.33202.2616

برنامه، بررسی شود.

آزمون رگرسیون فرآیندی پر هزینه می‌باشد. درطی این آزمون لازم است تا تمام سناریوهای تست که قبلاً برروی قسمت‌های برنامه آزمایش شده‌اند، دوباره مورد آزمون قرار گیرند. این موارد آزمون به صورت یک مجموعه درمی‌آیند که با هربار اعمال تغییرات در برنامه، تعداد این مجموعه بزرگ‌تر می‌گردد. بنابراین توسعه دهندگان برنامه به منظور کاهش هزینه‌های وارد شده به سخت‌افزار و متخصصین خود، نیاز دارند تا به کاهش هزینه‌ها و زمان این فرآیند پرهزینه بپردازند.

روترمل و همکاران[۱] بیان کرده‌اند که دو روش برای کاهش هزینه‌های آزمون رگرسیون وجود دارد. این دو روش شامل به‌حداقل رساندن مجموعه‌ی موارد آزمون رگرسیون و تکنیک‌های الویت‌بندی مجموعه موارد آزمون رگرسیون می‌باشند، که این تکنیک‌ها اطلاعات را از برنامه اصلی، برنامه‌ی اصلاح شده و مجموعه موارد آزمون می‌گیرند.

در تکنیک به حداقل رساندن مجموعه‌ی موارد آزمون رگرسیون، به انتخاب یک زیرمجموعه از مجموعه‌ی اولیه‌ی موارد آزمون پرداخته می‌شود، به‌شرطی که این زیرمجموعه تمام آن خطاهایی از برنامه را که مجموعه‌ی موارد آزمون اولیه آشکار می‌کرد را آشکار کند. به این ترتیب هزینه و زمان اجرای آزمون رگرسیون کاهش می‌یابد. در تکنیک الویت‌بندی موارد آزمون رگرسیون، موارد آزمون براساس یک‌سری از معیارهای تعیین شده الویت‌بندی و رتبه‌دهی می‌گردند و ابتدا موارد آزمون با الویت بالاتر اجرا می‌گردند، تا زمانی‌که به شرط حداکثر هزینه و زمان برسیم. در این مقاله روش جدیدی برای کاهش موارد آزمون رگرسیون مبتنی برروش بهینه سازی معلم- دانش آموز ارائه می‌شود. این روش از دو فاز آموزش و یادگیری روی مجموعه آزمون تشکیل شده و بر اساس پارامترهای مختلف پیاده‌سازی شده است. این روش، جواب قطعی را به ما نمی‌دهد بلکه جوابی نزدیک به جواب بهینه را تولید می‌کند. ادامه مقاله به این صورت است که در بخش بعد مسئله تحقیق بیان می‌شود. در بخش سوم، مهمترین تحقیقات انجام شده در زمینه کاهش موارد آزمون رگرسیون بررسی شده‌اند. در بخش چهارم روش پیشنهادی و جزئیات آن ارائه می‌گردد. نتایج آزمایش‌ها و مقایسه‌های انجام شده در بخش پنجم مورد بررسی و تحلیل قرار می گیرند. این مقاله در نهایت در بخش ششم جمع بندی و نتیجه گیری می‌شود.

۲- بیان مسئله

هدف ما در این تحقیق، کاهش هزینه و زمان اجرای آزمون رگرسیون به وسیله‌ی حذف موارد آزمون تکراری و هم پوشا است. علاوه بر این می‌خواهیم یک زیرمجموعه از موارد آزمون رگرسیون را به نحوی انتخاب کنیم که همان تأثیر آشکارسازی خطای مجموعه‌ی اولیه را در برنامه، داشته باشد. اولین تعریف رسمی به حداقل رساندن مجموعه موارد آزمون رگرسیون را هارولد و همکاران[۲] به این‌صورت ارائه داده‌اند که:

فرض کنید P یک برنامه باشد و P’ نسخه‌ی اصلاح شده‌ی آن برنامه باشد،T مجموعه موارد آزمون رگرسیون باشد که برنامه‌ی P را مورد آزمون قرار می‌دهد. هدف یافتن T’ یک زیر مجموعه از مجموعه‌ی مواردآزمون اولیه ($T' \subseteq T$) می‌باشد به‌طوری که هر خطایی که با T برروی P’ آشکار می‌شود با T’ نیز برروی P’ آشکار گردد.

۳- پیشینه‌ی تحقیق

تکنیک‌هایی که برای این هدف مورد استفاده قرار گرفته‌اند را می‌توان به دو گروه کلی تقسیم‌بندی نمود. گروه اول تکنیک‌هایی هستند که مبتنی بر روش دو مرحه‌ای معرفی شده توسط هارولد و همکاران[۲] استوار بودند. گروه دوم، تکنیک‌هایی هستند که مبتنی بر الگوریتم‌های فراابتکاری و مبتنی بر جمعیت بوده و به انتخاب یک راه‌حل بهینه از میان راه‌حل‌های موجود می‌پرداختند. این انتخاب توسط الگوریتم فراابتکاری و با بررسی نحوه‌ی عملکرد آن الگوریتم در انتخاب راه‌حل نزدیک به بهترین راه‌حل، انجام می‌شود.

۳-۱- تکنیک‌های مبتنی بر روش دو مرحله‌ای

روش دو مرحله‌ای مبتنی بر شناخت تمام تأثیرات اعمال اصلاح در نرم‌افزار است. با توجه به این‌که هر اعمال اصلاح در برنامه ممکن است درسایر قسمت‌هایی از برنامه که مورد اصلاح قرار نگرفته‌اند هم تأثیراتی گذاشته باشد، لازم است تا پس از تست قسمت‌های اصلاح شده‌ی برنامه، مراحل زیر انجام گیرد:

مرحله یک- قسمتی‌هایی از برنامه که مورد به‌روز رسانی قرار نگرفته‌اند اما از این قسمت‌ها تأثیر پذیرفته‌اند مشخص گردند.

مرحله دو- موارد آزمونی که مرتبط با این قسمت‌ها می‌باشند، در مجموعه‌ی موارد آزمون رگرسیون کاهش یافته آورده شوند.

با توجه به تعریف روش دو مرحله‌ای، تکنیک‌های مختلفی به منظور برآورده‌کردن مرحله‌ی اول یعنی شناخت قسمت‌های مشکوک به تحت تأثیر بودن از قسمت‌های اصلاح شده‌ی برنامه، معرفی گردیدند. شناخت این قسمت‌ها به طور قطعی چالش‌های فراوانی داشت و هر تکنیک با توجه به ابزاری که از آن استفاده می‌کند سعی در یافتن نتیجه‌ی قطعی مورد نظر دارد.

طبق بررسی که توسط چاراسیا و همکاران[۳] بر یک‌سری از این تکنیک‌ها انجام شده است، دو الگوریتم که در [۴،۵] آمده‌اند، مورد بررسی قرار گرفته‌اند. در هر دوی این تکنیک‌ها، از گراف کنترل جریان[5] استفاده شده است. این الگوریتم‌ها مسیر کنترل جریان منطقی برنامه را رسم می‌کنند و با توجه به آن، اقدام به ردگیری این گراف و درپی آن اقدام به تشخیص این قسمت‌های مشکوک می‌نمایند. این گراف باتوجه به سطح پایین کد، دستورات شرطی، حلقه‌ها و سایر دستورهای کنترل جریان رسم می‌گردد. با بررسی انجام گرفته، این روش‌ها در کاهش هزینه‌ها مؤثر بودند.

داهیا و همکاران[۶] اقدام به معرفی مفهومی به نام دیوار آتش[6] نمودند. دیوار آتش مفهومی که به بررسی دیاگرام کلاس‌های برنامه می‌پرداخت. کلاسی که اصلاحی در برنامه بر آن اعمال شده را در مرکز دیوار آتش قرار می‌داد و با توجه به وابستگی برنامه در سطح کلاس‌های آن، کلاس‌های وابسته را در دیوارآتش اضافه می‌نمود. سپس اقدام به انتخاب موارد آزمون مرتبط با این دیوار آتش می‌نمود و بقیه موارد آزمون را رها می‌کرد. به این ترتیب اقدام به کاهش موارد آزمون می‌نمود. مفهوم دیوار آتش ابتدا در سطح کلاس، مورد بررسی قرار گرفت. اما در بررسی‌های بعدی صورت گرفته مشخص شد مشکل انتخاب موارد آزمون هرز را دارد، چرا که اگر وابستگی در سطح کلاس‌ها بررسی گردد امکان دارد که تنها یک بخش از آن کلاس با دیوار آتش ارتباط داشته باشد. در این صورت اقدام به آزمون دوباره‌ی کل کلاس می‌شود و این هزینه‌ها را افزایش می‌دهد. لذا نسخه‌ای دیگر از دیوار آتش مبتنی بر سطح متد معرفی گردید.

همچنین در[۷،۸،۹] هم تکنیک‌هایی برای یافتن این قسمت‌های مشکوک معرفی شده‌اند که از ویژگی‌های منحصر به‌فرد برنامه‌ها استفاده کرده‌اند. به‌طور مثال از ویژگی‌هایی که برنامه‌های پایگاه داده‌ای دارد و تغییر وضعیتی که هر گزارش بر روی وضعیت کلی پایگاه داده ایجاد می‌کند. یا از ویژگی‌های برنامه‌های تحت وب استفاده شده است و یا از ویژگی‌های ساختار منحصر به فرد که مخصوص زبان C می‌باشند، استفاده شده تا راحت‌تر این ردگیری انجام گیرد. همچنین مفهوم به نام اسلایس‌بندی[7] برنامه مطرح گردیده است. در اسلایس‌بندی، برنامه براساس یک پارامتر خاص قطعه‌بندی می‌گردد و قسمت‌هایی از برنامه که در ارتباط با آن هستند در اسلایس مربوطه آورده می‌شوند. به این‌صورت اقدام به حداقل‌سازی برنامه می‌گردد تا حداقل‌سازی موارد آزمون تسهیل گردد.

۳-۲- تکنیک‌های مبتنی بر الگوریتم‌های فراابتکاری[8] و جمعیت[9]

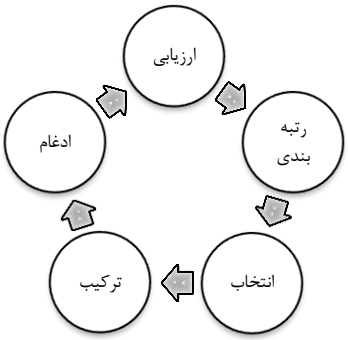
الگوریتم‌های فراابتکاری یا فرا تکاملی یا فرااکتشافی نوعی از الگوریتم‌ها هستند که برای یافتن پاسخ بهینه یا نزدیک به بهینه به کارمی‌روند. روش‌ها والگوریتم‌های بهینه سازی به دو دسته دقیق و الگوریتم‌های تقریبی تقسیم می‌شوند. الگوریتم‌های دقیق قادر به یافتن بهترین جواب به صورت دقیق‌اند اما درمورد مسائل بهینه‌سازی سخت کارایی ندارند. زمان آنها متناسب با ابعاد مسئله به صورت نمایی افزایش می‌یابد. الگوریتم های تقریبی قادر به یافتن جواب‌های خوب (نزدیک به بهینه) در زمان حل کوتاه، برای مسائل بهینه سخت هستند. الگوریتم‌های تقریبی به سه دسته ابتکاری، فراابتکاری و فوق ابتکاری تقسیم بندی می‌شوند. دو مشکل اصلی الگوریتم‌های ابتکاری گیر افتادن آنها در نقاط بهینه محلی و همگرایی زودرس به این نقاط است. الگوریتم‌های فراابتکاری برای حل مشکل الگوریتم‌های ابتکاری ارائه شده اند. در واقع الگوریتم‌های فراابتکاری یکی از انواع الگوریتم‌های بهینه سازی هستند که دارای راهکار برون رفت از نقاط بهینه محلی هستند. معیارهای مختلفی برای طبقه بندی این الگوریتم‌ها وجود دارد:

- معیاراول، مبتنی برجمعیت و مبتنی بر یک جواب: الگوریتم‌های مبتنی بر جمعیت درحین جستجو، یک جمعیت از جواب‌ها را درنظر می‌گیرند. این الگوریتم‌ها در حین جستجو یک جواب را انتخاب

[5] control-flow graph
[6] fire wall
[7] slicing
[8] Meta-heuristic-algorithms
[9] population

می‌کنند.

- معیار دوم، با حافظه و بدون حافظه: برخی از الگوریتم‌های فراابتکاری فاقد حافظه می‌باشند .به‌این معنا که این الگوریتم‌ها از اطلاعات بدست آمده درحین جستجو استفاده نمی‌کنند. الگوریتم تبرید شبیه‌سازی شده نمونه ای از این الگوریتم-هاست. در حالیکه برخی از الگوریتم‌ها نظیر جستجوی ممنوعه از حافظه استفاده می‌کنند.
- قطعی و احتمالی: یک الگوریتم فرا ابتکاری قطعی نظیر جستجوی ممنوعه، مسئله را با استفاده از تصمیمات قطعی حل می‌کند، اما در مورد الگوریتم‌های فراابتکاری احتمالی نظیر تبرید شبیه سازی شده یکسری قوانین احتمالی درحین جستجو مورد استفاده قرار می‌گیرد. از میان الگوریتم‌های شناخته شده فرا ابتکاری برپایه جمعیت می‌توان الگوریتم‌های تکاملی مثل الگوریتم ژنتیک، بهینه سازی کلونی مورچه‌ها، کلونی زنبور عسل و روش بهینه‌سازی ذرات را نام برد.



شکل ۱- فرآیند تکامل طبیعی

الگوریتم‌های فرا ابتکاری که مبتنی بر جمعیت می‌باشند، همان‌طورکه در شکل ۱ آمده‌است، از پنج مرحله‌ی عمده تشکیل شده‌اند که این مراحل عبارتند از ایجاد جمعیت اولیه با یک مجموعه‌ی تصادفی از جواب‌ها و ارزیابی آنها، مقایسه و رتبه‌بندی و انتخاب بهترین راه‌حل‌ها، ترکیب جواب‌های بدست آمده با شبیه‌سازی فرایند طبیعی و به وسیله‌ی الگوریتم موجود، ادغام نتایج و ارزیابی آنها و تولید و جایگزینی نسل جدید و بازگشت به مرحله‌ی اول.

الگوریتم‌های فراابتکاری دسته‌ای از روش‌های مبتنی برجمعیت هستند که در حل مسائل پیچیده‌ی بهینه‌سازی و مسائلی که مبتنی بر جستجوی جواب بهینه می‌باشند موفقیت‌های بسیاری داشته‌اند. در ادامه به بررسی روش‌های ارائه شده برای حل مسئله تحقیق توسط محققین که مبتنی بر این الگوریتم‌ها بوده‌اند، می‌پردازیم. کاندیل و همکاران[۱۰] یک روش مبتنی بر خوشه‌بندی[10] ارائه کردند که با توجه به چندین معیار، به خوشه‌بندی مجموعه موارد آزمون رگرسیون موجود، اقدام می‌نماید. این خوشه‌بندی از انتخاب موارد آزمون هرز و هم‌پوشانی موارد آزمون در مجموعه‌ی ثانویه جلوگیری می‌کند، چرا که موارد آزمونی که دارای پوشش خطاها با همپوشانی بالا می‌باشند در یک خوشه قرار گرفته و در هر خوشه تا حد امکان برازنده‌ترین موارد، انتخاب می‌گردند. به این ترتیب اقدام به کاهش هم‌پوشانی و کاهش موارد آزمون می‌گردد. در [۱۱]، پژوهشگران با استفاده از الگوریتم ژنتیک[11] اقدام به حل مسئله‌ی انتخاب راه‌حل بهینه از مجموعه موارد آزمون موجود کردند و جستجوی مربوط به آن را توسط جمعیتی که توسط الگوریتم ژنتیک و با جهش‌های آن به وجود می‌آید، انجام دادند. در الگوریتم ژنتیک می‌توان از ویژگی رهگیری نسل‌ها برای حذف موارد زائد و هم‌پوشا استفاده نمود.

نایاک و همکاران[۱۲] با استفاده از یک روش مبتنی بر الگوریتم بهینه‌سازی زنبورعسل[12] و استفاده از قوانین فازی[13] ، تیاگی و مالهوترا[۱۳] با استفاده با استفاده از یک روش چند مؤلفه‌ای مبتنی بر الگوریتم بهینه‌سازی ازدحام ذرات[14]، و سوری و همکاران[۱۴] با استفاده از روشی مبتنی بر الگوریتم بهینه سازی کلونی مورچگان[15] و تعریف توابع برازندگی و محاسبه‌ی هزینه برای هر یک از راه‌حل‌های پیشنهادی توسط الگوریتم‌ها، اقدام به انتخاب یک یا چند مجموعه از جواب‌ها به عنوان بهترین مجموعه‌ی کاهش یافته نمودند. در مقاله [۱۵] چگونگی استفاده از الگوریتم هوش مبتنی بر بهینه سازی ذرات به منظور اولویت بندی موارد آزمون در تست رگرسیون تحلیل و بررسی شده است. از الگوریتم فراابتکاری کرم شبتاب به منظور اولویت بندی موارد آزمون، در مقاله[۱۶] مورد توجه قرار گرفته است و نتایج قابل قبولی با به کارگیری این روش بدست آمده است. الگوریتم بهینه سازی مگس گل و میوه از دیگر الگوریتم‌های بهینه سازی فراابتکاری است که به منظور اولویت بندی

[10] clustering
[11] genetic algorithm
[12] honey bee optimization algorithm
[13] fuzzy rule base
[14] particle swarm optimization
[15] ant colony optimization

موارد آزمون رگرسیون به کار گرفته شده است[۱۷]. در مقاله [۱۸] از الگوریتم فرااکتشافی جدیدی بنام بهینه سازی ازدحام ذرات کوانتم-رفتار بهبود یافته به منظور اولویت بندی، انتخاب و کاهش موارد آزمون استفاده شده است. طبق بررسی‌های انجام شده در[۲۲-۱۹] بر روی تحقیقات و مقاله‌هایی که بین سال‌های ۲۰۰۹ تا ۲۰۲۱ و در زمینه‌ی کاهش مجموعه موارد آزمون رگرسیون در جهت کاهش هزینه و زمان آزمون رگرسیون صورت گرفته است و مبتنی بر الگوریتم‌های فراابتکاری و اکتشافی بوده‌اند، روش‌های که براساس جستجوی حداقل هزینه و زمان از میان راه‌حل‌های پیشنهادی توسط الگوریتم مبتنی بر جمعیت بوده‌اند با حدود ۲۵٪ از کل تحقیقات در رتبه‌ی اول علاقه‌مندی قرار دارند. از طرف دیگر روش‌هایی که به دنبال کشف زیرمجموعه با حداکثر پوشش خطای موجود بودند با ۱۸٪ از کل تحقیقات در رتبه‌ی دوم جای دارند.

روش پیشنهادی این مقاله مبتنی بر الگوریتم معلم-دانش آموز یا TLBO[۱۶] [۲۳] می‌باشد. این روش یک جستجوی مبتنی بر جمعیت را شکل می‌دهد و از میان راه‌حل‌هایی که جمعیت تولید می‌نماید اقدام به محاسبه‌ی تابع برازندگی که در اینجا پیشنهاد شده است، می‌نماید. سپس اقدام به جایگزینی جمعیت جدید و سپس تکرار نموده تا زمانی که به شرط پایان برسد و سپس بهترین راه‌حل را با استفاده از محاسبه‌ی بهترین برازندگی به عنوان یک زیر مجموعه از مجموعه‌ی اولیه معرفی می‌نماید. در زیر مزایای الگوریتم TLBO نسبت به سایر روش‌های بهینه‌سازی فرا ابتکاری را بررسی می‌کنیم:

۱. سادگی: یکی از مزیت‌های اصلی الگوریتم TLBO، سادگی پیاده‌سازی آن است. این الگوریتم قدرتمندی است که با استفاده از قوانین ساده مبتنی بر آموزش و یادگیری، به جستجوی بهینه‌سازی می‌پردازد. این سادگی در پیاده‌سازی الگوریتم و همچنین در تنظیم پارامترها، به کاربر اجازه می‌دهد به سرعت الگوریتم را در مسائل خود به کار بگیرد.

۲. عملکرد قوی: الگوریتم TLBO به عنوان یک الگوریتم بهینه‌سازی فراابتکاری، عملکرد قوی در مسائل پیچیده و چندمعیاره داشته و می‌تواند بهینه‌سازی بهتری را نسبت به روش‌های دیگر ارائه دهد. این الگوریتم از ترکیب عملگرهای تدریس و یادگیری استفاده می‌کند تا به یک تعامل اجتماعی بین اعضای جمعیت برسد و از تجربیات همدیگر استفاده کند تا بهینه‌سازی را انجام دهد.

۳. جستجوی موازی: الگوریتم TLBO قابلیت جستجوی موازی را دارد. با تقسیم جمعیت به چند زیرمجموعه و انجام عملیات تدریس و یادگیری در هر زیرمجموعه، الگوریتم می‌تواند به صورت موازی اجرا شود و به سرعت بهینه‌سازی را انجام دهد. این ویژگی منجر به کاهش زمان اجرا و افزایش سرعت جستجوی می‌شود.

۴. استفاده از عملگرهای تدریس و یادگیری: الگوریتم TLBO از عملگرهای تدریس و یادگیری الهام گرفته است که توانایی بهبود جمعیت را دارند. این عملگرها، تعامل اجتماعی و تبادل اطلاعات بین اعضای جمعیت را افزایش می‌دهند و باعث بهبود کیفیت جستجو می‌شوند. این عملگرها شامل عملگر تدریس (Teaching Phase) و عملگر یادگیری (Learning Phase) است که قابلیت استفاده از تجربیات اعضای بهتر جمعیت را دارند و این امکان را فراهم می‌کنند تا جمعیت به صورت تدریجی بهینه شود.

۵. قابلیت تطبیق و انعطاف‌پذیری: الگوریتم TLBO قابلیت تطبیق و انعطاف‌پذیری بالایی دارد. با توجه به خواص محیط بهینه‌سازی و نیازهای مسئله، می‌توان پارامترها و قوانین الگوریتم را تنظیم کرد و آن را به بهترین شکل ممکن برای مسئله مورد نظر سازگار کرد. این ویژگی باعث می‌شود الگوریتم در حل مسائل متنوع و متفاوت با کارایی بالا عمل کند. به طور خلاصه، مزایای الگوریتم TLBO نسبت به سایر روش‌های بهینه‌سازی فراابتکاری عبارتند از: سادگی پیاده‌سازی، عملکرد قوی در مسائل پیچیده، قابلیت جستجوی موازی، استفاده از عملگرهای تدریس و یادگیری، قابلیت تطبیق و انعطاف‌پذیری. با این ویژگی‌ها، الگوریتم TLBO می‌تواند به عنوان یک ابزار قدرتمند در حل مسئله کاهش موارد آزمون رگرسیون مورد استفاده قرار گیرد. بنابراین در ادامه به معرفی روش پیشنهادی که مبتنی بر الگوریتم یادگیری معلم و دانش آموز است، می‌پردازیم.

۴- روش پیشنهادی

یک مسئله مهم در تست نرم افزار، این است که تست همه ترکیبات موارد آزمون ازنظر هزینه و زمان ممکن نیست. بنابراین درتست نرم افزار مسئله انتخاب بهینه موارد آزمون جهت صرف جویی در هزینه و زمان مطرح است. شناسایی

[16] Teaching-learning based optimization algorithm

موارد آزمون خوب نیاز به تعریف معیارهایی برای آزمون می باشد. برای نمونه می‌توان پوشش مسیر، پوشش انشعاب، پوشش شرطی و پوشش خطا را نام برد. همان گونه از نام معیارهای مطرح شده مشخص است، داده آزمون باید به نوعی تولید و از فضای مسئله انتخاب شود که به ازای اجرای برنامه با داده تولیدی معیارمورد نظر بطور کامل پوشش داده شود. در زیر مشکلات عمده برای انجام تست رگرسیون ذکر شده است:

۱. با اجرای تست رگرسیون مداوم، مجموعه‌ی موارد آزمون نسبتا بزرگ می‌شود. با توجه به محدودیت‌های زمانی و بودجه، نمی‌توان آزمون دوباره‌ی کامل را اجرا کرد.

۲. با کوچک‌سازی مجموعه‌ی موارد آزمون رگرسیون رسیدن به حداکثر پوشش خطا همچنان به عنوان یک چالش باقی می‌ماند.

اما تست رگرسیون به معنی تست برنامه نرم‌افزاری شما در زمانیست که یک کد تغییر می‌کند و هدف این تست، حصول اطمینان از این موضوع است که کد جدید شما، قسمت‌های دیگر نرم‌افزار را تحت تاثیر قرار نمی‌دهد. یکی دیگر از مزایای تست این است که ناخواسته مستندات برنامه را که نشان دهنده چگونگی عملکرد برنامه است، تهیه می کنیم. با استفاده از تست می‌توانیم ورودی‌های نادرست را به برنامه داده و نتیجه تست را مشاهده کنیم. شاید برای اغلب برنامه نویسان تازه کار تست کردن کار سخت و دشواری باشد و گمان کنند که این کار برای برنامه کوچکشان لازم نیست ولی اگر برنامه‌های حتی کوچک خود را تست کنید چنانچه در آینده قصد توسعه آن را داشته باشید با مشکلات کمتری روبه رو خواهید شد.

دراین قسمت سناریوی مورد استفاده دراین تحقیق که براساس الگوریتم معلم-دانش آموز (TLBO) [۲۴] است را برای افزایش سرعت تست نرم افزار مبتنی بر کاهش مجموعه آزمون تست، ارائه می‌نماییم. در زیر دلایل مناسب بودن الگوریتم معلم-دانش آموز برای حل مسئله کاهش موارد آزمون رگرسیون را بر شمرده‌ایم.

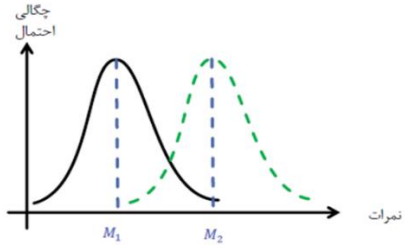
اول اینکه، در مسئله کاهش موارد آزمون رگرسیون، هدف اصلی این است که تعداد موارد آزمون را به حداقل برسانیم، در حالی که خطای پیش‌بینی مدل رگرسیون را نیز در حداقل ممکن نگه داریم. با کاهش تعداد موارد آزمون، هزینه و زمان مورد نیاز برای اجرای آزمون را نیز کاهش می‌دهیم. در اینجا، TLBO به عنوان یک الگوریتم بهینه‌سازی، می‌تواند عملکرد مناسبی داشته باشد. دومین دلیل، عملکرد گروهی مناسب این الگوریتم است. TLBO بر مبنای یادگیری از تدریس از طریق تعامل بین دو گروه اصلی، یعنی گروه معلمان و گروه دانش‌آموزان عمل می‌کند. این الگوریتم با استفاده از این تعامل، توانایی جستجوی همزمان و موازی را دارد. در مسئله کاهش موارد آزمون رگرسیون، این ویژگی می‌تواند منجر به بهبود سرعت جستجو و توانایی الگوریتم در پیدا کردن جواب بهینه شود. دلیل سوم، تنوع جمعیتی این الگوریتم است. TLBO با تعامل بین اعضای جمعیت، تنوع جمعیت را حفظ می‌کند. این ویژگی می‌تواند در کاهش موارد آزمون مفید باشد، زیرا تنوع در موارد آزمون به ما کمک می‌کند تا پوشش متنوعی از رفتارهای سیستم را در نظر بگیریم و به صورت جامع تست‌ها را انجام دهیم. دلیل چهارم، قابلیت تطبیق پارامترهاست، TLBO قابلیت تطبیق پارامترها را دارد و می‌توان آن را بر اساس خصوصیات و ویژگی‌های مسئله کاهش موارد آزمون رگرسیون بهینه‌سازی کرد. این ویژگی به ما امکان می‌دهد تا الگوریتم را بر حسب نیازهای خاص مسئله تنظیم کنیم و عملکرد بهینه را بدست آوریم.

الگوریتم بهینه سازی معلم-دانش آموز نیز مشابه سایر روش های بهینه سازی موجود، یک الگوریتم برگرفته از طبیعت و مبتنی بر جمعیت است و بر اساس تأثیر یک معلم بر روی یادگیری در کلاس درس کار می‌کند. این الگوریتم از یک جمعیتی از جواب‌ها برای دست‌یابی به جواب کلی استفاده می‌نماید. جمعیت به عنوان گروهی از یادگیران یا دانش آموزان یک کلاس در نظر گرفته می‌شود. یک معلم تلاش می‌کند تا با آموزش به دانش آموزان، سطح دانش کلاس را افزایش دهد و دانش آموز به نمره یا رتبه خوبی مطابق با توانایی خودش، دست یابد. در حقیقت یک معلم خوب کسی است که دانش آموز خود را به سطح دانش خود یا نزدیک به خود برساند.

معلم یک شخص با دانش بالا در جامعه بوده که علم خود را با دانش آموزان خود تقسیم می‌کند، به طوری که بهترین جواب (بهترین عضو جمعیت) در همان تکرار به عنوان معلم عمل می‌کند. اما لازم است به این نکته اشاره شود که دانش آموزان مطابق با کیفیت آموزش ارائه شده توسط معلم و وضعیت شاگردان حاضر در کلاس، دانش کسب می‌کنند. علاوه بر این دانش آموزان از تعامل متقابل بین خودشان که به وضعیت‌شان کمک می‌کند، استفاده می‌کنند.

الگوریتم بهینه‌سازی TLBO براساس تأثیر یک معلم بر روی خروجی دانش آموزان در یک کلاس است و به طور کلی در یک کلاس، معلم فردی تعیین می‌شود که دارای مقدار بهتری از لحاظ تابع تناسب است و سطحی بالاتری نسبت به دانش آموزان دارد و می‌تواند دانش آموزان را از دانش خود سهیم نماید. یک معلم خوب، یک میانگین بهتر برای دانش آموزان تولید می‌کند. در هر مرحله و تکرار، معلم کسی است که بهترین فرد کلاس باشد و بهترین مقدار تابع هدف را دارد. البته در هر مرحله ممکن است معلم تغییر نماید.

این الگوریتم جهت استفاده در امور مهندسی گزینه بسیار مناسبی است. در این الگوریتم از فرآیند آموزش و یادگیری که در کلاس درس اتفاق می‌افتد، الهام گرفته شده است. به این ترتیب که وقتی معلم، درس را ارائه می‌دهد در نهایت ارزیابی کرده و دانش آموزان نیز نمره‌ای را کسب می‌نمایند. اگر نمودار شکل (۲) نمودار توزیع احتمال نمرات افراد باشد، منحنی سمت چپ مربوط به کلاس اول و منحنی سمت راست مربوط به کلاس دوم یا معلم دیگر می‌باشد. با مقایسه سطح نمرات این دو کلاس در می‌یابیم که معلم دوم موفق‌تر عمل کرده‌است چرا که میانگین سطح کلاس آن که M2 می‌باشد، از میانگین کلاس اول یعنی M1 بالاتر است.

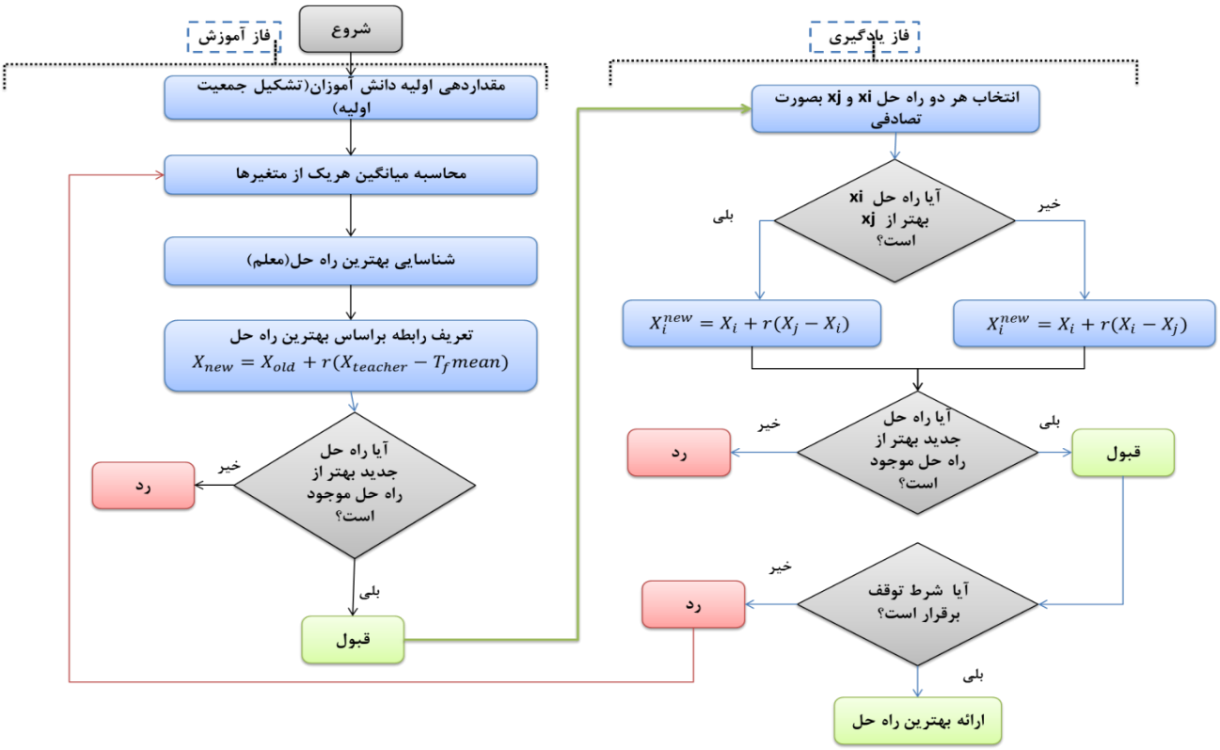


شکل ۲- نمودار گاوسی توزیع چگالی احتمال نمرات دانش‌آموزان

در این نمودار برای نمایش توزیع نمرات از توزیع گاوسی استفاده شده است. اما می‌توان از هر توزیع دیگری نیز استفاده نمود. طبق رابطه چگالی احتمال توزیع گاوسی میانگین مقادیری مثل M1 و M2 می‌باشد. در این الگوریتم به پراکندگی نمرات یا واریانس پرداخته نمی‌شود و مهم‌ترین نکته همان میانگین نمرات دانش آموزان کلاس است. الگوریتم بهینه سازی مبتنی بر یادگیری و آموزش(معلم-دانش‌آموز) یا TLBO دارای دو مرحله یا دو فاز اصلی است که عبارتند از:

۱. مرحله معلم یا فاز آموزش
۲. مرحله دانش آموز یا فاز یادگیری

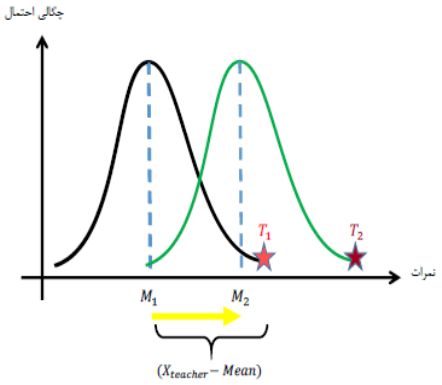
در شکل (۳) فلوچارت مربوط به الگوریتم یادگیری معلم و دانش‌آموز آمده است.



شکل ۳- فلوچارت الگوریتم معلم-دانش آموز

۴-۱- فاز معلم یا آموزش

در فاز اول، معلم سعی میکند تا میانگین کلاس را به سطح خود برساند و سطح دانش آموزان در این مرحله به سمت معلم تغییر می‌نماید(شکل ۴). در این الگوریتم معلم از بین دانش آموزان انتخاب می‌شود، یعنی کسی که اطلاعاتش از بقیه بیشتر و بهتر باشد به عنوان معلم انتخاب خواهد شد.



شکل ۴- نمودار تغییرات میانگین در فاز آموزش

نمونه T1 به عنوان معلم کلاس انتخاب شده و سعی می‌کند میانگین سطح کلاس یعنی $M1$ را به سطح خودش برساند. اما در واقعیت این امر امکان پذیر نیست که همه دانش آموزان به سطح معلم برسند بلکه نهایت به سطح میانگین جدید $M2$ خواهند رسید.

۴-۱-۱- مدل ریاضی برای فاز معلم

مدل ریاضی معلم در رابطه (۱) نشان داده شده است. اندازه گام جابجایی برابر است با: $X_{teacher} - T_f * Mean$ که در این رابطه $X_{teacher}$ همان میانگین مطلوب یا بهترین عضو کلاس می‌باشد و Mean نیز میانگین اعضای جمعیت در دور فعلی است. بنابراین رابطه ریاضی برای فاز معلم به این صورت خواهد شد:

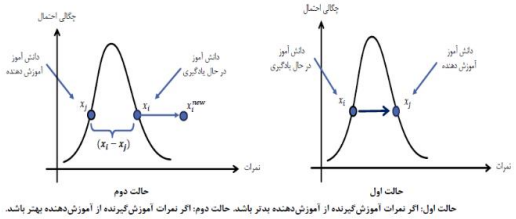
$$X_{new} = X_{old} + r(X_{teacher} - T_f mean) \qquad (۱)$$

که در آن r یک بردار تصادفی بین صفر و یک بوده که میزان موفقیت یک دانش آموز در درک مطالب یاد داده شده توسط استاد یا معلم را نشان می‌دهد. بدین صورت که اگر r صفر باشد یعنی دانش آموز از مطالبی که معلم به او آموزش داده چیزی یاد نگرفته است و اگر r برابر یک باشد یعنی تمام مطالب معلم را فراگرفته است. البته مقدار r بین صفر و یک است و بسته به عدد تصادفی ایجا شده می‌تواند یادگیری متفاوت باشد. برای درک بهتر r را می‌توان ضریب هوشی دانش آموز نیز در نظر گرفت.

همچنین Tf یا ضریب معلم نشان دهنده ضریب موفقیت معلم است که می‌تواند عدد ۱ یا ۲ باشد. اگر Tf معلم یک باشد، یک یادگیری معمولی را خوهیم داشت ولی اگر این ضریب ۲ باشد شتاب یادگیری بیشتر خواهد شد. Tf را می‌توان قدرت بیان یک معلم در نظر گرفت. به طور کلی هدف از تزریق r و Tf در رابطه بالا ایجاد تنوع جمعیتی است تا فضای جستجو به صورت تصادفی توسط افراد جمعیت جستجو شود.

۴-۲- مرحله دانش آموز یا فاز یادگیری

فاز دانش آموز بعد از فاز معلم اجرا می‌شود و دانش آموزان می‌توانند از یکدیگر نیز آموزش ببینند و بر روی یکدیگر تأثیر بگذارند و این تعامل باعث می‌شود که سطوح دانش‌آموزان ارتقاء پیدا نماید. این فاز که نام دیگر آن فاز یادگیری می‌باشد همان طور که در شکل (۵) مشخص شده است دانش آموزان با تعامل و بحث و گفتگو با یکدیگر سعی در افزایش سطح دانش هم دارند.



شکل ۴- فاز دانش‌آموزان

با توجه به شکل (۵)، دو دانش آموز به صورت تصادفی از بین جمعیت انتخاب می‌شوند و در آن دانش آموز اول یعنی X_i می‌خواهد از دانش آموز دوم یعنی X_j آموزش ببیند. بسته به میزان نمره این دو دانش آموز دو حالت برای اثر پذیری دانش آموز X_i و دانش آموز X_j بوجود خواهد آمد:

۱. حالت اول: اگر نمرات دانش آموز در حال یادگیری از آموزش دهنده بدتر باشد.

۲. حالت دوم: اگر نمرات دانش آموز در حال یادگیری از دانش آموز آموزش دهنده بهتر باشد.

۴-۲-۱- حالت اول: اگر نمرات دانش آموز در حال یادگیری از دانش آموز آموزش دهنده بدتر باشد

در این حالت یک دانش آموز ضعیف X_i می خواهد از یک دانش آموز با نمرات بهتر X_j آموزش ببیند یعنی تا حد ممکن می‌خواهد فاصله خودش را با همکلاسی‌اش کم کند. در نتیجه مشابه فاز معلم برای افزایش تنوع جمعیتی به این رابطه ایده آل یک ضریب تصادفی r اضافه می‌نماییم. رابطه این حالت به صورت زیر تعریف می‌شود:

$$X_i^{new} = X_i + r(X_j - X_i) \quad (۲)$$

۴-۲-۲- حالت دوم: اگر نمرات دانش آموز در حال یادگیری از آموزش دهنده بهتر باشد.

همان طور که در شکل (۵) مشخص است در این حالت دانش آموز در حال یادگیری که همان X_i می‌باشد از دانش آموز در حال آموزش یعنی X_j دارای نمرات بهتری می باشد. بدین منظور برای بهبود موقعیت X_i باید تا جای ممکن از X_j دور شود یا فاصله بگیرد. بنابراین عکس حالت قبلی اتفاق خواهد افتاد یعنی:

$$X_i^{new} = X_i + r(X_i - X_j) \quad (۳)$$

۴-۳- ورودی استاندارد الگوریتم

ورودی این الگوریتم یک مجموعه از موارد آزمون رگرسیون است. هر مورد آزمون در برنامه، خطا یا خطاهایی را کشف می‌نماید. همچنین هر مورد آزمون برای اجرا شدن هزینه و زمانی را در پی دارد که این باعث می‌شود در انتخاب موارد آزمون به دنبال کم هزینه‌ترین راه‌حل موجود باشیم. همان‌طور که در جدول ۱ آمده‌است ورودی استاندارد الگوریتم به صورت یک ماتریس است که سطرهای آن موارد آزمون است و ستون‌های آن، خطاهای پوشش داده شده توسط کل موارد آزمون و ستون آخر هزینه و زمان هر مورد آزمون است. در خانه‌های این ماتریس و به صورت باینری در صورتی که مورد آزمونی خطایی را پوشش دهد ۱ و در غیر این صورت ۰ قرار می‌گیرد. بنابراین بهترین موارد آزمون مطلوب، آزمون‌هایی هستند که بیشترین خطاهای موجود را شناسایی کرده و زمان اجرای کمتری نیز دارند.

جدول ۱- قالب ورودی الگوریتم

خطا / مورد تست	خطای اول	خطای دوم	خطای سوم	...	خطای نهم	خطای دهم	زمان
مورد تست اول	۱	۰	۱		۱	۰	۷
مورد تست دوم	۰	۱	۰	...	۰	۰	۳
مورد تست سوم	۰	۱	۰		۰	۰	۵
مورد تست چهارم	۰	۰	۰		۱	۰	۳
مورد تست پنجم	۱	۰	۰	...	۰	۱	۵
مورد تست ششم	۰	۰	۰		۰	۰	۶
مورد تست هفتم	۰	۰	۰	...	۰	۰	۳
مورد تست هشتم	۰	۰	۱		۰	۱	۵

۴-۴- تابع برازندگی

برای محاسبه‌ی میزان برازندگی هر یک از راه‌حل‌های پیشنهاد شده توسط الگوریتم، لازم است تا تابعی وجود داشته باشد که میزان برازندگی را به صورت عددی محاسبه و برای مقایسه با سایر راه‌حل‌ها ذخیره کند. در کاهش مجموعه موارد آزمون رگرسیون دو متغییر متضاد با هم وجود دارند.

- میزان پوشش خطا که مطلوب است تا حداکثر گردد.
- میزان زمان و هزینه که مطلوب است تا حداقل گردد.

و این دو در تضاد قرار دارند. لذا در تابع برازندگی پیشنهادی، میزان پوشش خطا از رابطه‌ی زیر محاسبه می‌گردد که برابر جمع or منطقی تمام موارد آزمون است. یعنی اگر از میان مجموعه‌ی انتخاب شده‌ی کاهش یافته‌ی ما، حداقل یکی از موارد آزمون آن خطا را پوشش دهد، آن خطا برابر ۱ قرار می‌گیرد. سپس این تعداد بر تعداد کل خطاها تقسیم و درصد گرفته می‌شود.

$$Fault\_coverage = 100 * \frac{\left|\bigcup_{tj=1}^{\{f(t)\}}\right|}{m} \quad (۴)$$

در صورتی که مجموع تعداد خطای پوشش داده شده توسط هر مورد آزمون بر کل تعداد خطاها، برابر با یک باشد، پوشش کامل برای آن مورد آزمون وجود دارد. میزان زمان و هزینه‌ی هر راه‌حل هم از رابطه‌ی زیر و از جمع زمان موارد آزمون انتخاب شده به دست می‌آید:

$$Executation\ Time = \sum_{tj=1} ei \quad (۵)$$

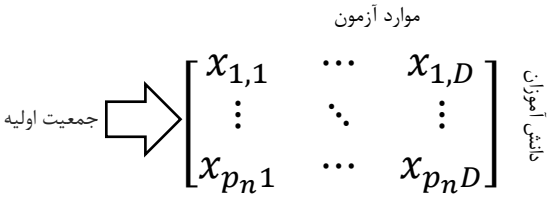
در رابطه فوق، هر مورد آزمون یک زمان اجرای ej دارد. به منظور حداکثرسازی مقدار تابع برازندگی، کسری تشکیل داده و مقدار رابطه (۴) را در صورت و مقدار رابطه (۵) را در مخرج محاسبه می‌نماییم.

۴-۵. نحوه‌ی اجرای الگوریتم بر روی ورودی استاندارد

الگوریتم پیاده‌سازی شده‌ی پیشنهادی طیف متغیر از ورودی با تعداد موارد آزمون متغیر و تعداد خطاهای متغیر را دریافت می‌کند. همچنین در الگوریتم یک‌سری پارامتر متغیر برای مشاهده‌ی نتیجه‌ی تاثیر آن‌ها بر جواب نهایی

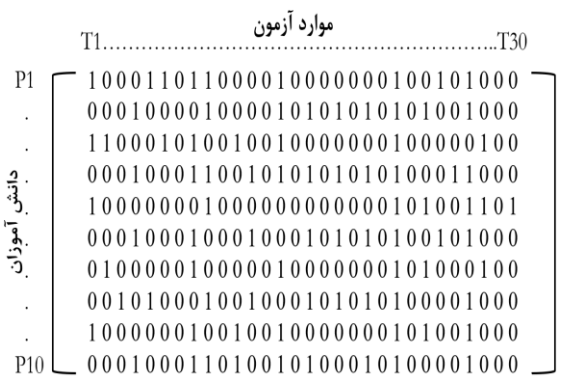
وجود دارد. از جمله تعداد تکرار[1] الگوریتم که مشخص می‌کند الگوریتم TLBO چند تکرار روی جمعیت انجام دهد و ما این مقدار را معمولاً روی ۱۰۰۰ تنظیم می‌کنیم. همچنین تعداد دانش‌آموزان که برابر تعداد جمعیت ما می‌باشد و با کم و زیاد کردن تعداد دانش‌آموزان می‌توان مقایسه نمود که جواب نهایی بهتر یا بدتر می‌شود.

برنامه پس از دریافت ورودی اقدام به تولید یک جمعیت تصادفی اولیه و به صورت ماتریس مطابق آن‌چه که در شکل۶ آمده است، می‌نماید. در این ماتریس تعداد ستون‌ها برابر تعداد موارد آزمون مجموعه‌ی ورودی می‌باشد و مقدار ۰ یا ۱ در خانه‌ی مربوط به آن، مشخص می‌کند که آن مورد آزمون انتخاب می‌شود یا خیر. تعداد سطرها برابر تعداد جمعیت دانش‌آموزان می‌باشد.



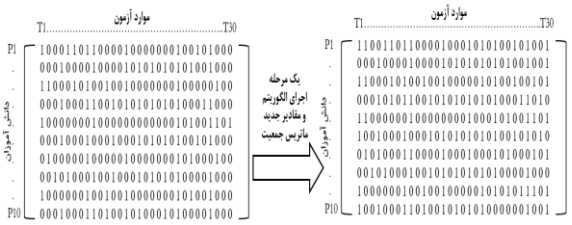
شکل ۵- ماتریس جمعیت اولیه

لذا اگر مجموعه‌ی موارد آزمون ورودی در برنامه شامل ۳۰ مورد آزمون و تعداد دانش‌آموزان برابر ۱۰ باشد ماتریس تصادفی جمعیت اولیه به صورت شکل (۷) تشکیل می‌شود.



شکل ۶- ماتریس تصادفی جمعیت اولیه

که پس اجرای فاز آموزش و فاز یادگیری دانش آموز جدیدی تولید می‌گردد و در صورتی که با محاسبه برازندگی این دانش آموز بهتر از دانش آموز قبلی باشد جایگزین می‌گردد طبق شکل (۸) این روند اجرا می‌گردد تا شرط پایان الگوریتم برآورده شود.



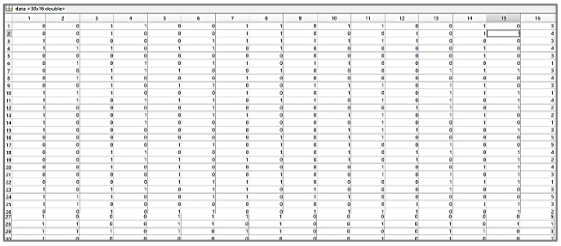
شکل ۷- روند اجرای یک مرحله از الگوریتم بر روی ماتریس دانش‌آموزان

۵-شبیه سازی و نتایج

در این قسمت نتایج عددی روش پیشنهادی در مقایسه با الگوریتم‌های هم راستا مطرح شده است. این مقایسه برای نمونه‌هایی در شرایط و چالش‌های مختلف صورت گرفته است. تمامی آزمایش‌های این پژوهش با نرم افزار متلب در سیستم عامل ویندوز ۷ پیاده سازی و روی یک لپ تاپ Dell با پردازنده مرکزی ۲ هسته ای ۲ گیگاهرتز و حافظه داخلی ۲ گیگابایت انجام شده است. در قسمت قبل، رویکرد انتخاب موارد آزمون مبتنی بر الگوریتم معلم- دانش آموز را ارائه نمودیم. در این قسمت نتایج حاصل از اجرای الگوریتم معلم- دانش آموز را در انتخاب بهینه موارد آزمون، ارائه می‌دهیم.

۵-۱- ورودی برنامه

ورودی استاندارد یک ماتریس از موارد آزمون و خطاها در برنامه و زمان اجرای هر مورد آزمون در ستون آخر می‌باشد. مطابق شکل(۹) این ماتریس به صورت تصادفی با مقادیر صفر و یک پر شده است و زمان هریک به صورت تصادفی از مقادیر یک تا پنج پر شده است. از این ورودی برای ارزیابی روش‌های مرجع نیز استفاده شده‌است تا شرایط برابر باشد.



شکل ۸- ورودی برنامه

۵-۲. نتایج عددی الگوریتم

برای مقایسه‌ی تأثیر تعداد دانش آموزان در جواب نهایی، تعداد جمعیت(Npop) را در مرحله تغییر داده و نتایج را ذخیره کردیم. همچنین به دلیل وجود بعضی از پارامترهای

[1] iteration

تصادفی در رابطه‌ی ریاضی الگوریتم مثل Tf و r، اقدام به اجرای ۱۰ بار از برنامه برای هر مقدار Npop نموده و میانگین هر ۱۰ بار را به عنوان نتیجه‌ی خروجی درنظر گرفتیم. نتایج برای حالتی که تعداد موارد آزمون ۳۰، تعداد خطا ۱۵، با جمعیتی برابر با Npop=۳۰ و iteration=1000 بصورت جدول ۲ حاصل شده است.

جدول ۲- موارد آزمون پیشنهادی و زمان هر راه‌حل

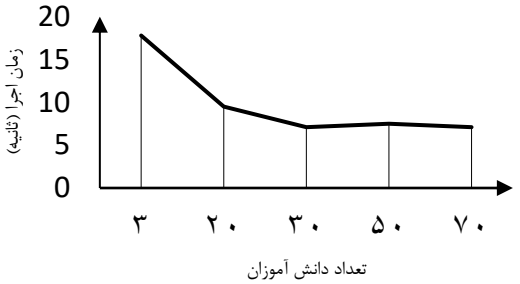
تعداد دفعات اجرا	ترتیب پیشنهادی الگوریتم معلم- دانش آموز	زمان اجرای ترتیب انتخابی موارد آزمون
1	11 10 19	6
2	29 26 6	5
3	11 13 26	8
4	11 10 26	7
5	11 10 19	7
6	11 10 19	7
7	11 10 26	7
8	26 10 1	6
9	11 10 19	6
10	11 10 19	7

همچنین با تغییر تعداد دانش‌آموزان به ۳ و ۲۰ و ۳۰ و ۵۰ و ۷۰، مقدار زمانی برای هریک از پارامترها به صورت متوسط ده بار اجرا محاسبه گردید و در جدول۳ آمده است. همان طور که در این جدول مشخص است با افزایش تعداد دانش‌آموزان تا مقدار مشخصی که این‌جا ۳۰ است، زمان جواب نهایی به صورت قابل ملاحظه‌ای کاهش می‌یابد.

جدول ۳- میانگین زمان ده بار اجرا برای تعداد متغیر دانش‌آموزان

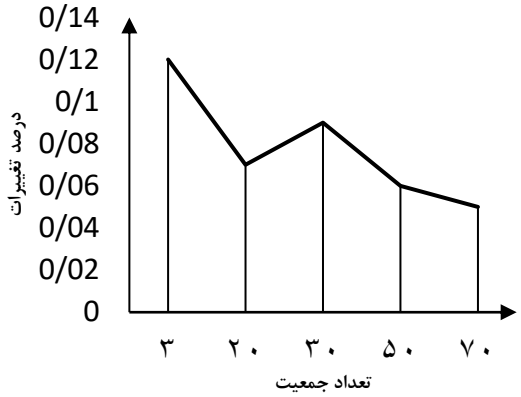
تعداد موارد‌آزمون	تعداد خطا	تعداد جمعیت(دانش آموز)	تعداد دور	میانگین زمان اجرا مجموعه ی کاهش یافته
30	15	3	1000	17.8
30	15	20	1000	9.5
30	15	30	1000	7.2
30	15	50	1000	7.5
30	15	70	1000	7.1

این نتایج به صورت نمودار شکل (۱۰) نیز نشان داده شده است. در این شکل دیده می‌شود که پس از مقدار ۳۰ برای تعداد دانش آموزان، کاهش بیشتری نداریم. پس این مقدار، برای تعداد دانش آموزان مقدار مناسبی است.



شکل ۱۰- میانگین زمان ده بار اجرا بر حسب تغییر تعداد دانش آموزان

شکل (۱۱) درصد تغییرات جواب بهینه در الگوریتم پیشنهادی نسبت به اندازه جمعیت را نشان می‌دهد. با وجود این‌که الگوریتم TLBO یک الگوریتم فرا اکتشافی تقریبی می‌باشد شکل (۱۱) مشخص می‌کند که این درصد تا حد زیادی نزدیک به صفر است.



شکل ۱۱- درصد تغییرات جواب بهینه با افزایش جمعیت

۵-۳- مقایسه نتایج با روش‌های دیگر

به منظور مقایسه‌ی الگوریتم پیشنهادی با سایر روش‌ها از سه روش مرجع اولویت بندی موارد آزمون که در [۱۶] معرفی گردیده است استفاده نمودیم. همچنین برای شرایط برابر در نتایج از همان دیتاست ورودی روش پیشنهادی برای این روش استفاده نمودیم. همچنین برای مقایسه، دو الگوریتم فراابتکاری دیگر، یعنی روش مبتنی بر بهینه سازی زنبور عسل و منطق فازی ارائه شده توسط نایاک و همکاران [۱۲] و همچنین روش مبتنی بر بهینه سازی ازدحام ذرات ارائه شده توسط تیاگی و مالهوترا [۱۳] را استفاده کرده‌ایم. سه روش اولویت بندی موارد آزمون رگرسیون [۱۶] به ترتیب شامل موارد زیر می‌باشند:

- روش مرتب‌سازی تصادفی[1] که به انتخاب تصادفی موارد آزمون و اضافه نمودن آن‌ها در مجموعه‌ی کاهش یافته اقدام می‌کند تا شرط حداکثر پوشش خطا برآورده شود.
- روش بدون مرتب‌سازی[2] که اقدام به انتخاب موراد آزمون از ابتدای ماتریس ورودی می‌نماید تا زمانی که شرط برآورده شود.
- روش مرتب‌سازی معکوس[3] که اقدام به انتخاب موارد آزمون از انتهای ماتریس ورودی می‌کند تا زمانی که شرط برآورده شود.

[1] random ordering
[2] no ordering
[3] reverse ordering

نتیجه‌ی حاصل توسط الگوریتم بدون مرتب‌سازی، الگوریتم مرتب‌سازی معکوس، و الگوریتم مرتب‌سازی تصادفی به ترتیب در جدول ۴، جدول ۵، و جدول ۶ آمده است.

جدول ۴- نتایج الگوریتم بدون مرتب سازی

موارد آزمون	خطاهای پوشش داده شده توسط هر مورد آزمون	زمان اجرای مورد آزمون
T1	3 4 7 8 10 11 14	4
T2	3 7 8 12 14 15	2
T3	1 6 7 8 9 10 11 12	3
T4	1 2 3 5 6 8 14	3
T5	14	3
T6	2 4 6 7 9 10	1
T7	3 4 5 8 13 14 15	3

جدول ۵- نتایج الگوریم مرتب‌سازی معکوس

موارد آزمون	خطاهای پوشش داده شده توسط هر مورد آزمون	زمان اجرای مورد آزمون
T30	1 11 13	2
T29	1 2 3 5 7 8 13 14	3
T28	1 2 5 6 8 11 13 15	1
T27	1 5 6 7 8	4
T26	3 5 6 9 10 11 12 15	2
T25	1 2 12 14 15	4
T24	1 2 3 6 7 8 10	5
T23	1 3 4 6 9 13 3	3

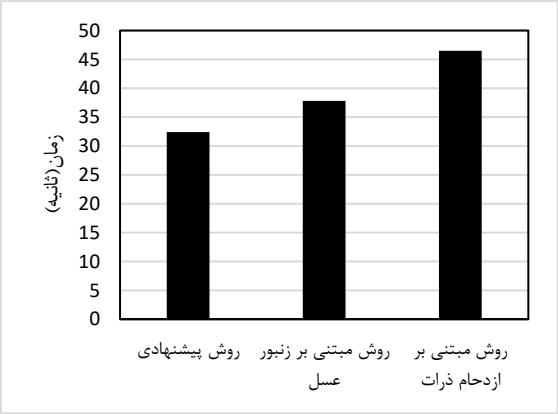
جدول ۶- نتایج الگوریتم مرتب‌سازی تصادفی

موارد آزمون	خطاهای پوشش داده شده توسط هر مورد آزمون	زمان اجرای مورد آزمون
T8	1 6 7 8 9 10 11 12	3
T2	14	3
T23	3 4 5 8 13 14 15	3
T14	1 4 6 7 10 11 13 14	2
T15	1 2 3 7 10 13 14 15	1

نتایج عددی از جواب‌های بدست آمده توسط این سه الگوریتم و مقایسه‌ی آن‌ها با میانگین نتایج روش پیشنهادی در جدول ۷ آمده‌است و مشخص کننده‌ی بهبود قابل توجه روش پیشنهادی در زمان است. در سه روش موجود مؤلفه‌ی پوشش حداکثر خطا به تنهایی مورد توجه قرار گرفته و الگوریتم تا زمان رسیدن به حداکثر پوشش خطا به کار خود ادامه می‌دهد. دراین بین از انتخاب و مقایسه‌ی سایر راه‌حل‌های ممکن که همان پوشش حداکثر را دارا بوده اما از هزینه و زمان کمتری نیز برخوردارند، چشم‌پوشی می‌کند. لذا استفاده از روش‌هایی که مانند روش پیشنهادی از یک الگوریتم فرابتکاری و مبتنی بر جمعیت برای تولید و مقایسه‌ی جایگشت‌های ممکن از میان مجموعه‌ی موارد آزمون اولیه استفاده می‌کنند، باعث می‌شود که به انتخاب شایسته‌ترین راه حل براساس معیارهای شایستگی موجود برسیم.

جدول ۷- مقایسه‌ی زمان سه روش با میانگین زمان روش پیشنهادی

Test Suite	No Ordering	Reverse Ordering	Random	معلم- دانش آموز
Test suit 1	21	23	12	6.6



شکل۱۲- مقایسه‌ی زمان روش پیشنهادی با دو الگوریتم فرا ابتکاری دیگر

شکل (۱۲) میزان بهبود زمان روش پیشنهادی با دو روش مبتنی بر الگوریتم‌های فراابتکاری یعنی الگوریتم بهینه‌سازی زنبورعسل [۱۲] و الگوریتم بهینه‌سازی ازدحام ذرات [۱۳] را نشان می‌دهد. همان طور که از نتایج مشخص است روش پیشنهادی دارای زمان بهتری است یعنی سریع تر به جواب می‌رسد. دلیل تفاوت بیشتر در زمان روش مبتنی بر الگوریتم بهینه‌سازی ذرات این است که این الگوریتم با وجود سر راست بودن و این که در حداقل زمان به حداکثر نتیجه‌ی بهینه که می‌تواند کشف کند می‌رسد، اما اگر تعداد جمعیت ذرات کمتر از یک مقدار باشد یا اندازه‌ی گام‌هایی که ذرات به سمت یکدیگر حرکت می‌کنند کوچکتر از یک مقدار نباشد، الگوریتم ممکن است جواب بهینه را رد کرده و انتخاب نمی‌نماید. همچنین روش مبتنی بر الگوریتم بهینه‌سازی زنبورعسل از نظر زمان اجرا، مقداری نزدیک به روش پیشنهادی دارد اما در این روش به دلیل اینکه در تابع برازندگی تنها از مؤلفه‌ی پوشش حداکثری خطا به عنوان عامل مؤثر در انتخاب راه‌حل‌های بهینه استفاده شده است، باعث می‌شود در مواردی که می‌توان با مقدار کمی کمتر شدن پوشش خطا، به مقدار قابل توجهی زمان و هزینه‌ی اجرای موارد آزمون را کاهش داد، این روش نقص پیدا می‌کند. این درحالی است که در روش پیشنهادی با وارد کردن این مؤلفه‌ی زمان در فرمول تابع برازندگی و یک مصالحه بین آن و مؤلفه‌ی پوشش حداکثری خطا، این

شرایط در نظر گرفته شده‌است. این مورد و همچنین عملکرد مناسب الگوریتم معلم و دانش‌آموز در تکامل و انتخاب جمعیت باعث درصد بهبود بهتر روش پیشنهادی نسبت به دو روش دیگر شده است.

۶- جمع بندی نتیجه‌گیری

الگوریتم‌های فرا ابتکاری کاربردهای گوناگونی دارند[۲۶,۲۵]. روش پیشنهادی این مقاله مبتنی بر الگوریتم فراابتکاری معلم-دانش آموز است. این الگوریتم جزو الگوریتم‌های فرا ابتکاری می‌باشد که در حل مسائل مربوط به بهینه‌سازی با بدست آوردن یک جواب تقریبی و نزدیک به بهینه، اقدام به حل مسائل مختلف مهندسی می‌نماید. این روش مانند دیگر الگوریتم‌های فرا ابتکاری مبتنی بر جمعیت، به‌صورت تصادفی از یک نقطه شروع کرده و تا رسیدن به بهینه‌ی سراسری کار خود را ادامه می‌دهد. الگوریتم معلم-دانش آموز برای حل مسائل بهینه‌سازی پیچیده طراحی شده است، اما پیچیدگی محاسباتی نسبتاً بالایی دارد. این به معنای اجرای زمان‌بر الگوریتم و نیاز به منابع محاسباتی بیشتر است. در مقابل، روش‌های سنتی و غیر فراابتکاری ممکن است بتوانند به صورت سریعتر و با استفاده کمتر از منابع محاسباتی به نتایجی برسند. الگوریتم معلم دانش آموز دارای پارامترهایی است که باید به درستی تنظیم شوند. انتخاب نادرست پارامترها می‌تواند درکارایی الگوریتم و نتایج بهینه تأثیر منفی بگذارد. تنظیم پارامترها نیازمند آزمون و ارزیابی تکراری الگوریتم بر روی مسائل مختلف است که سربار زمانی و محاسباتی دارد. در روش‌های سنتی و قطعی، پارامترهای ثابتی استفاده می‌شود و نیاز به تنظیم آنها نیست. مانند هر الگوریتم بهینه سازی دیگری، برای اجرای الگوریتم معلم-دانش آموز، نیاز به داده‌های آموزشی وجود دارد. جمع‌آوری داده‌های آموزشی و آماده‌سازی آن‌ها ممکن است طولانی و زمان‌بر باشد. روش‌های سنتی، ممکن است نیازی به داده‌های آموزشی نداشته باشند یا نیاز به حجم کمتری داشته باشند. به طور کلی، استفاده از راهکار معلم-دانش آموز نسبت به روش‌های قطعی ممکن است سربارها و هزینه‌هایی داشته باشد. با این حال، باید توجه داشت که الگوریتم معلم-دانش آموز می‌تواند در مسائل پیچیده و با ابعاد بالا مثل کاهش موارد آزمون رگرسیون که برای روش‌های سنتی و قطعی دشوار است، عملکرد بهتری داشته باشد. این روش در مقایسه‌ی عددی نتایج به دست آمده نسبت به سه روش مرجع بهتر عمل نمود. همچنین کارایی بهتری نسبت به دو روش فرا ابتکاری قبلی دارد. علت این برتری به دلیل وجود تابع برازندگی مبتنی بر حداکثر سازی پوشش خطا و حداقل‌سازی زمان است. علاوه براین، الگوریتم پیشنهادی عملکرد مناسبی در پیشرفت و رشد و تولید جمعیت جدید بر اساس معیار تابع برازندگی دارد. همچنین به دلیل مشاهده‌ی عملکرد مناسب الگوریتم‌های تکاملی در روش‌های ارائه شده قبلی و روش پیشنهادی این مقاله، در حل مسائل بهینه سازی و حداقل‌سازی مجموعه راه حل‌ها، پیشنهاد می‌شود که به منظور حل این دست مسائل بهینه‌سازی همچنان به الگوریتم‌های فراابتکاری اعتماد گردد. همچنین با بهینه‌سازی‌هایی بر روی تابع برازندگی و نحوه انتخاب بهترین راه حل و تنظیم پارامترهای الگوریتم فراابتکاری[۲۷] می‌توان کارایی اینگونه الگوریتم‌ها را بهبود داد.

مراجع

[1] G. Rothermel, R.J. Untch and C. Chu. “Prioritizing Test Cases for Regression Testing.” *IEEE Transactions on Software Engineering* 27. no.10 (2001): 929-948.

[2] M.J. Harrold, J.V. Ronne and C. Hong. “Empirical Studies of TestSuite Reduction.” *Software Testing, Verification and Reliability* 12. no.4 (2002): 219-249.

[3] V.Chaurasia, Y.Chauhan, and K. Thirunavukkarasu. “A survey on test case reduction techniques.” *International Journal of Science and Research (IJSR)*, 2014.

[4] Q. Wang, S. Jiang and Y. Zhang. “An approach to generate basis path for programs with exception-handling constructs.” *In IACSIT Press, International Conference on Computer Science and Information Technology (ICCSIT)* ,2011.

[5] R.P. Mahapatra, M. Mohan and A. Kulothungan. “Effective tool for test case Execution time reduction.” *In IACSIT, International Symposium on Computing*, 2011.

[6] S. Dahiya, R.K Bhatia, and D. Rattan. “Regression test selection using class, sequence and activity diagrams.” *IET Software* 10. no.3 (2016): 72-80.

[7] F. Haftman, D. Kossmann, and E. Lo. “A framework for efficient regression tests on database applications.” *The VLDB Journal* 16 (2007): 145-164.

[8] M. Al-Refai, “Improving Model-Based Regression Test Selection. ” *in Proceedings of the ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MoDELS’18). Austin, TX, USA*, 2018.

[9] E. Engström, P. Runeson, and M. Skoglund. “A systematic review on regression test selection techniques.” *Information and Software Technology* 52. no.1 (2010): 14-30.

[10] P. Kandil, S. Moussa, and N. Badr. “Cluster-based Test Cases Prioritization and Selection Technique for Agile Regression Testing.” *Journal of Software: Evolution and Process* 29. no. 6 (2017): e1794.

[11] X.Y. Ma, B.K. Sheng, and C.Q.Ye. “Test-suite reduction using genetic algorithm.” *Advanced Parallel Processing Technologies: 6th International Workshop, APPT 2005, Hong Kong, China, October 27-28, 2005. Proceedings 6*. Springer Berlin Heidelberg, 2005.

[12] S. Nayak, C. Kumar, S. Tripathi, N. Mohanty, and V. Baral. “Regression test optimization and prioritization using Honey Bee optimization algorithm with fuzzy rule base.” *Soft Computing* 25 (2021): 9925-9942.

[13] M. Tyagi and S. Malhotra. "Test case prioritization using multi objective particle swarm optimizer." *International Conference on Signal Propagation and Computer Technology (ICSPCT 2014)*. IEEE, 2014.

[14] B. Suri and S. Singhal. “Test case selection & prioritization using ant colony optimization.” *International Conference on Advanced Computing, Communication and Networks, Chandigarh*. vol. 194. 2011.

[15] B.A.K.R., ba-quttayyan, H. Mohd, and Y. Yusof. “a critical analysis of swarm intelligence for regression test case prioritization.” *Journal of Theoretical and Applied Information Technology 100.* no.12 (2022): 3997-4025.

[16] M. Khatibsyarbini, M. A. Isa, D.N. Jawawi, H.N.A. Hamed, and M.D.M. Suffian. “Test case prioritization using firefly algorithm for software testing.” *IEEE Access* 7 (2019): 132360-132373.

[17] Vedpal, H. Tanwar, N. Chauhan, and M. Khanna. “Test case prioritization using a Hybrid Chaotic Flower-fruit fly optimization algorithm with multiple objectives.” *Multimedia Tools and Applications* 83. no. 10 (2024): 28395-28418.

[18] A. Bajaj, A. Abraham, S. Ratnoo, and L.A. Gabralla. “Test case prioritization, selection, and reduction using improved quantum-behaved particle swarm optimization.” *Sensors* 22. no.12 (2022): 4374.

[19] E. Engström, P. Runeson, and M. Skoglund. “A systematic review on regression test selection techniques.” *Information and Software Technology* 52. no.1 (2010): 14-30.

[20] O. Dahiya, K. Solanki. “A systematic literature study of regression test case prioritization approaches.” *International Journal of Engineering & Technology* 7. no. 4 (2018): 2184-2191.

[21] M. Khatibsyarbini, M.A. Isa, D.N. Jawawi, and R. Tumeng. “Test case prioritization approaches in regression testing: A systematic literature review.” *Information and Software Technology* 93 (2018): 74-93.

[22] M. Hasnain, I. Ghani, M.F. Pasha, and S.R. Jeong. “Ontology-based regression testing: a systematic literature review.” *Applied Sciences* 11. no. 20 (2021): 9709.

[23] R.V. Rao, V.J. Savsani, and D.P. Vakharia. “Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems.” *Computer-aided design* 43. no. 3 (2011): 303-315.

[24] F. Zou, D. Chen, and Q. Xu. “A survey of teaching–learning-based optimization.” *Neurocomputing* 335 (2019): 366-383.

[25] A. Ebrahimi, A. Hajipour, and H. Tavakoli. "Localization in IoT by using Fractional Order Chaotic Particle Swarm Algorithm Optimization." *Journal of Modeling in Engineering* 18. no. 60 (2020): 157-168. (in Persian)

[26] H. Bigdeli, S.M.S. Mirdamadi, and J. Tayyebi. "A meta-heuristic method for maximum capacity path interdiction problem with multiple attackers." *Journal of Modeling in Engineering* 20. no. 70 (2022): 133-146. (in Persian)

[27] E. Shadkam, and M. Ghayoor. "A new hybrid method DSM for parameter setting of meta-heuristic algorithms." *Journal of Modeling in Engineering* 19.65 (2021): 161-180. (in Persian)