مجله علمی-پژوهشی «علوم و فناوری‌های پدافند غیرعامل»
سال دوم، شماره ۲، تابستان ۱۳۹۰: ص ۹۰- ۸۳

تولید خودکار دادگان آزمون به کمک شبکه عصبی

رضا ترکاشون[۱*]، محمدرضا کنگاوری[۲]

۱-کارشناس ارشد، ۲- استادیار، دانشکده مهندسی کامپیوتر، دانشگاه علم و صنعت ایران

(دریافت: ۱۳۹۰/۰۲/۱۸، پذیرش: ۱۳۹۰/۶/۱۳)

چکیده

یکی از مراحل مهم آزمون نرم‌افزار شئ‌گرا، آزمون مستقل اشیا است. آزمون مستقل اشیا با دو مشکل روبه‌رو است: اولاً شئ مورد فراخوانی ممکن است روش‌هایی از اشیا دیگر را فراخوانی کند و در نتیجه بررسی مستقل آن ممکن نباشد. ثانیاً روش‌های فراخوانی شده ممکن است زمان‌بر باشند و باعث شوند آزمون شئ مورد نظر طولانی شود. یک راه‌حل برای رفع دو مشکل فوق، استفاده از اشیا جاعل است. اشیا جاعل روش‌های مورد فراخوانی را شبیه‌سازی کند. اشیا جاعلی که تاکنون معرفی شده‌اند مبتنی بر جدول هستند و خود از مشکلات زمان‌بر بودن و مهمتر از آن عدم توانایی در شبیه‌سازی دقیق روش‌ها رنج می‌برند. از سوی دیگر دادگان آزمون کم می‌باشد و تولید خودکار موارد آزمون با حداکثر میزان پوشش مسیرهای اجرایی در برنامه‌های مورد آزمون مورد توجه بوده است. این مقاله شامل دو بخش پیشنهادی است، در بخش اول مقاله با استفاده از شبکه‌های عصبی عملکرد توابع خطی درون برنامه‌ها شبیه‌سازی می‌شود. همچنین با به‌کارگیری الگوریتم ژنتیک، بهترین زیر مجموعه از ورودی‌ها برای آموزش شبکه عصبی را از بین مجموعه بزرگی از ورودی‌ها که به صورت تصادفی ایجاد شده‌اند، در بخش دوم تعیین می‌شود. در این تحقیق یک شئ جاعل مبتنی بر شبکه عصبی پیشنهاد می‌گردد که هر دو مشکل اشیا جاعل مبتنی بر جدول را رفع کند. آزمایش‌ها روی توابع ریاضی، منطقی و گسسته نشان می‌دهد که روش پیشنهادی در هر دو بخش، عملکرد مناسبی داشته‌اند.

کلیدواژه‌ها: شبکه‌های عصبی، آزمون مستقل اشیاء، شیء جاعل، الگوریتم ذوب فلز، الگوریتم ژنتیک

Automatic Test Data Generation Using Artificial Neural Networks

R. Torkashvan[1*], M. Kangavari[2]
Computer Engineering Department, Iran University of Science and Technology (IUST)
(Received: 05/08/2011, Accepted: 09/04/2011)

Abstract

Independent object test is one of the important steps in the object-oriented software test. This kind of test is faced with two problems: firstly, the object which is under test may call methods of other objects and therefore, the independent test of the object becomes impossible. Secondly, the called methods may be time-consuming and as a result the test of the object takes a long time. In order to overcome these problems, a useful method is to use the faked object which simulates the called methods. Faked objects are usually implemented using a table. This table-based implementation results in different problems such as time-consuming table search operation, and more importantly, inability to exact simulation of called methods. Besides, test samples are rare and therefore automatic generation of test samples which span all the code paths within a method has become a challenging problem. In this paper, a new artificial neural network-based faked object is proposed which solves the two above-mentioned problems. This paper contains two proposed sections: in the first section, the operation of linear functions which are used in programs is simulated. In the second section, the best set of input parameters which are needed to train the artificial neural network of faked object is determined optimally using genetic algorithm. The superiority of the proposed methods is confirmed using different experiments for mathematical, logical and discrete functions.

Keywords: Neural Networks, Independent Object Test, Faked Object, Simulated Annealing, Genetic Algorithm

* Corresponding author E-mail: torkashvan@iust.ac.ir

۱. مقدمه

یکی از مراحل مهم و حساس و در عین حال پرهزینه در فرآیند تولید نرم‌افزار شئ‌گرا، آزمون نرم‌افزار است. روش‌های مختلفی برای آزمون نرم‌افزار شئ‌گرا وجود دارند که هر کدام جنبه متفاوتی از نرم‌افزار را مورد ارزیابی قرار می‌دهند. این روش‌ها را می‌توان به سه دسته کلی آزمون وظایف، آزمون رفتار و آزمون مستقل اشیا تقسیم کرد.

در آزمون وظایف، هدف اطمینان از انجام درست هر یک از وظایف برنامه شئ‌گراست. انجام این آزمون به دلیل توزیع یک وظیفه در بین اشیا مختلف برنامه دشوار است. برای مثال اگر برنامه یکی از وظایف خود را به درستی انجام ندهد، نمی‌توان به آسانی تعیین کرد منشا خطا در کدام یک از اشیا برنامه است[۱].

دسته دیگری از روش‌های آزمون نرم‌افزار شئ‌گرا، سعی دارند درست بودن رفتار اشیا برنامه را بررسی کنند. منظور رفتار یک شئ، دنباله تغیر وضعیت‌های یک شئ است. ممکن است در یک شئ، هر یک از روش‌ها به تنهایی به درستی عمل کنند، اما ترتیب فراخوانی روش‌ها و در نتیجه ترتیب تغیر وضعیت‌های شئ به‌گونه‌ای باشد که رفتار مورد انتظار از شئ حاصل نشود. برای انجام آزمون رفتار اشیا مراحل زیر به ترتیب انجام می‌شود: ابتدا دیاگرام حالت اتمی[1] رسم می‌شود[۲].

دیاگرام اتمی حالت‌های مختلف یک شئ و تغییر حالت‌های آن شئ با فراخوانی هر روش را نشان می‌دهد. سپس برای اشیائی که در برگیرنده چند شئ داخلی هستند با کنار هم قرار دادن دیاگرام‌های حالت اتمی اشیاء داخلی، یک دیاگرام حالت ترکیبی[2] برای شئ در برگیرنده رسم می‌شود. در مرحله بعد، با توجه به دیاگرام حالت ترکیبی شئ، درخت فراخوانی روش‌های آن شئ رسم می‌شود. درخت فراخوانی، فراخوانی‌های تودرتوی ممکن از روش‌ها و تغییر حالت شئ به ازای آن فراخوانی‌ها را نشان می‌دهد[۲].

هر شاخه درخت فراخوانی متناظر با یک رفتار شئ می‌باشد. در مرحله پایانی، با پیمایش تمام شاخه‌های درخت از پایین به بالا و با ارزیابی روش‌های موجود در آن شاخه، رفتارهای مختلف شئ ارزیابی می‌شوند[۳].

یک نوع دیگر از روش‌های آزمون شئ‌گرا، در ارتباط با آزمون مستقل اشیا است. هدف از این آزمون، بررسی عملکرد یک شئ به‌طور مجزا و مستقل از سایر اشیا موجود در برنامه است. موضوعی که انجام این آزمون را دشوار می‌کند این است که شئ مورد آزمون ممکن است در انجام روش‌هایش از اشیا دیگر استفاده کند. این موضوع دو مشکل را در پی دارد:

اولاً ، اگر شئ مورد آزمون، وظایف خود را به درستی انجام ندهد، نمی‌توان با اطمینان گفت آن شئ دچار خطاست چون ممکن است خطا در شئ‌ای فراخوانی شده باشد.

ثانیاً اگر اجرای روش‌های شئ مورد استفاده زمان‌بر باشد، بررسی شئ مورد آزمون نیز زمان‌بر خواهد شد. برای رفع این مشکل، عملکرد روش‌های اشیا مورد استفاده را با شئ به نام شئ جاعل شبیه‌سازی می‌کنند[۳و۱].

از سوی دیگر تولید خودکار دادگان آزمون برای تعیین صحت عملکرد نرم‌افزار از چالش‌های مهم به حساب می‌آید. برای رسیدن به این هدف، خودکارسازی آزمون دو گام اساسی باید انجام شود: نخست، ورودی‌های مناسب برای برنامه تولید شود که منجر به طی شدن مسیرهای مناسب و متنوعی در برنامه شود. دوم الگویی از رفتار صحیح برنامه ایجاد شود تا معیاری برای تشخیص صحت عملکرد نرم‌افزار باشد.

ایجاد داده‌های ورودی آزمون و خروجی‌های مورد انتظار مورد توجه محققان بوده است. در این راستا، ایجاد تابع عملکرد برنامه‌ها با استفاده از شبکه‌های عصبی برای اولین بار در سال ۲۰۰۷ مطرح شد[۴]. استفاده از الگوریتم ژنتیک برای ایجاد ورودی‌های مناسب برای آزمون برنامه‌ها نیز بسیار مطرح بوده است[۴]. یکی از ساده‌ترین روش‌ها برای ایجاد داده‌های آزمون، جستجوی تصادفی می‌باشد. برای این منظور سعی می‌شود در بین ورودی‌های ممکن تعدادی به صورت تصادفی انتخاب شوند[۵].

بر اساس میزان پوشش مسیرهای اجرایی برنامه مورد آزمون، در یک فرایند تکاملی، به تدریج می‌توان بهترین مجموعه ورودی‌ها را ایجاد نمود[۷ و۶]. در عمل، فرایند تکاملی نتایج بهتری نسبت به روش‌های جستجوی تصادفی داشته‌اند[۵]. این گروه از روش‌های آزمون، به آزمون تکاملی مطابق با Wegener و Grochtmann ارجاع داده می‌شوند [۷]. آزمون تکاملی تکنیک خودکار ایجاد موارد آزمون بر مبنای استراتژی تکاملی[۶]، الگوریتم ژنتیک[۱۴]، برنامه‌نویسی ژنتیک[۴] و ذوب فلز[۸] می‌باشد.

در این مقاله، ابتدا با استفاده از دو روش شبیه‌سازی ذوب فلزات و الگوریتم ژنتیک نرخ تقسیم مجموعه دادگان به دو زیر مجموعه داده‌های آموزشی و آزمون اصلی بررسی خواهد شد. در هر دو روش پیشنهادی از معیار خطای رده‌بند مبتنی بر شبکه عصبی استفاده شده است. سپس با معرفی روش‌های موجود برای پیاده‌سازی شئ جاعل و بررسی اشکالات روش‌های موجود، روشی برای پیاده‌سازی شئ جاعل با استفاده از شبکه عصبی پیشنهاد می‌شود. سپس روش‌های پیشنهادی از نظر دقت و سرعت با روش‌های مشابه مقایسه خواهند شد.

در بخش دوم، به مروری بر کارهای انجام شده خواهیم پرداخت. بخش سوم به ارائه الگوریتم پیشنهادی و بررسی جزییات آن می‌پردازد و در بخش چهارم آزمایش‌ها و نتایج ارائه خواهد شد و در نهایت در بخش پنجم جمع‌بندی نهایی انجام می‌گیرد.

[1] Atomic Object State Diagram (AOSD)
[2] Composite Object State Diagram (COSD)

۲. مروری بر کارهای انجام شده

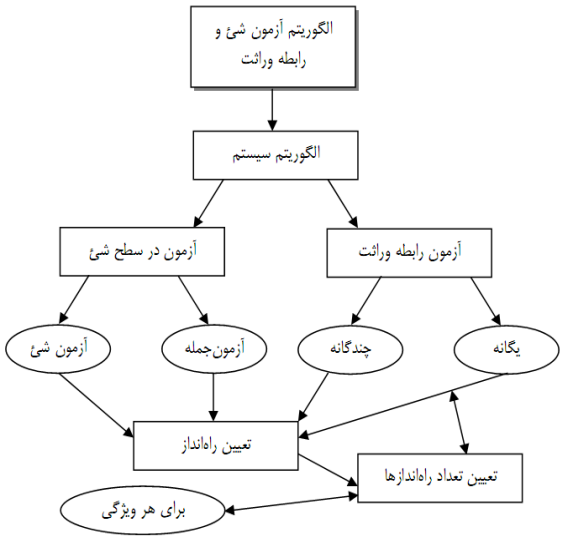
می‌توان با استخراج خودکار گراف، فراخوانی‌ها از متن برنامه مورد آزمون و حذف حلقه‌ها ساختار لایه‌ای برای برنامه را مشخص نمود. حلقه‌ها به‌واسطه فراخوانی‌های خودبازگشتی ایجاد می‌شوند. برای تشخیص حلقه‌ها در گراف فراخوانی می‌بایست لبه‌های برگشتی را تعیین نمود. در این راستا، الگوریتم‌هایی مطرح می‌باشد. با حذف حلقه‌ها و ایجاد گراف کمینه پوشا می‌توان ساختار سلسله مراتبی فراخوانی‌ها را برای آزمون رفتار اشیا مشخص نمود. البته این روش خودکار برای مواردی مورد استفاده است که برنامه آماده و در دسترس قرار دارد. بدین ترتیب می‌توان گره‌های درخت یا به عبارت دیگر روش‌ها را از برگ‌ها و سطح به سطح به سمت ریشه مورد بررسی و آزمون قرار داد.

در آزمون خودکار نرم‌افزار دو مسئله مطرح است: یکی تشخیص اجرای خطادار و دیگری تعیین محل و عامل خطا می‌باشد. می‌توان با استفاده از ترکیبی از شبکه‌های عصبی با الگوریتم‌های تکاملی و ذوب فلز، خروجی‌های مورد انتظار روش‌ها را بر اساس پارامترهای ورودی داده شده به روش و قبل از به اجرا در آوردن روش مشاهده و بررسی نمود. در ادامه این روش و چگونگی به‌کارگیری آن برای پیش‌بینی نتایج حاصل از اجرای روش‌ها ارائه شده است[۹ و۸].

در روش مبتنی بر مشخصه و مستندسازی سامانه، رفتار صحیح برنامه در قالب زبان صوری بیان می‌شود. روش‌های اکتشافی، نتایج تخمینی نادقیق برای یک مجموعه از ورودی‌های آزمون تولید می‌کند[۵]. در روش آماری از خواص آماری برنامه برای تولید اوراکل استفاده می‌شود. در اوراکل مبتنی بر نمونه، از یک نمونه برای تشخیص صحت عملکرد نرم‌افزار استفاده می‌شود. و داده‌ها به‌طور دستی تولید می‌شوند و با استفاده از شئ جاعل دستی شبیه‌سازی مسیرهای اجرایی مورد آزمون ایجاد می‌شوند[۷]. این کار دارای مشکلاتی از جمله صحت درستی شئ، پرهزینه بودن و زمان‌بر بودن می‌باشد. با استفاده از تولید داده به‌صورت خودکار مشکل زمان بر بودن و هزینه آزمون شئ جاعل حل شده است؛ زیرا بعضی از اشیاء به خاطر وابسته بودن به اشیاء دیگر زمان آزمون آنها طولانی می‌شود و هزینه زیادی صرف آزمون این اشیاء می‌گردد[۶].

۲-۱. راه‌انداز مسیر برای ایجاد شئ جاعل

راه‌انداز آزمون تابعی است که به‌صورت خودکار راه‌اندازهای آزمون را ایجاد، گردآوردی و به‌صورت پویا اجرا می‌کند. نحوه ایجاد راه‌اندازهای آزمون در شکل (۱) نشان داده شده است. برای هر شی، راه‌اندازهای آزمون مورد نیاز در سطح شی و رابطه‌های وراثت به تعداد کل راه‌انداز-های مورد نیاز اضافه می‌شوند. تخمین تعداد راه‌اندازهای آزمون مورد نیاز برای هدایت سامانه آزمون به مسئول آزمون در طراحی فرآیند آزمون کمک می‌کند[۱۰].



شکل۱. راه‌انداز مسیر

راه‌اندازهای آزمون در OOTA III برای برنامه‌های جاوا، C++ و C# در سطح شی، رابطها و رابطه وراثت ایجاد می‌شوند. برای هر نوع آزمون درون برنامه یک راه‌انداز جداگانه ایجاد می‌شود. می‌توان در محیط OOTA III برای آزمون یک روش مورد نظر و یا همه روش‌ها، به‌طور خودکار راه‌انداز آزمون ایجاد نمود.

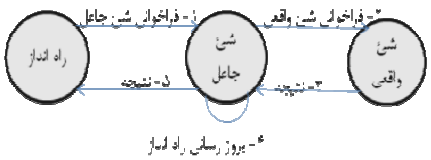
ابزار OOTA III در محیط NET تحت سامانه عامل XP قابل اجرا است. این ابزار برخط و مبتنی بر منو می‌باشد. این ابزار از چهار کلاس اصلی تشکیل شده است: کلاس شئ، کلاس رابط، کلاس ارث-بری و کلاس نتایج ورودی‌ها برای برنامه‌های مختلف متفاوت است اما خروجی‌ها استاندارد می‌باشند. خروجی شامل یک راه‌انداز آزمون برای دو مورد زیر می‌باشد:

- آزمون اشیاء و آزمون واسط
- ارث‌بری یگانه، چندگانه و چندسطحی.

از یک ارائه برای نگهداری کد لازم جهت ایجاد راه‌انداز استفاده می‌شود. برای ایجاد راه‌انداز خاص یک روش می‌بایست نام کلاس، نام روش، پارامترها و نام فایل کلاس مافوق توسط کاربر مشخص شود. جهت ایجاد راه‌انداز برای کلیه روش‌ها کاربر می‌بایست صرفاً نام سند را که نام کلاس در آن تعریف شده مشخص نماید[۲].

۲-۲. ایجاد شئ جاعل

مسیرها به طور عملی به کمک شئ‌ای به نام راه‌انداز مسیر به‌دست می‌آیند. در شکل (۲) یک راه‌انداز مسیر مشخص شده است. همان گونه که مشاهده می‌شود، راه‌انداز روشی از شئ جاعل را فراخوانی می‌کند. سپس شئ جاعل شئ واقعی را فراخوانی می‌کند و پارامترهای فراخوانی و نتیجه برگشتی را ذخیره می‌کند[۲].



شکل۲. شئ جاعل

اشیاء جاعل طوری تنظیم می‌شوند که مقادیر مورد استفاده و مورد نیاز برای شئ مورد آزمون را برگشت دهند. علاوه بر این، با استفاده از اشیایی به نام mock [۱۱]، می‌توان نشان داد که اشیاء جاعل به درستی مورد استفاده قرار می‌گیرند[۳].

هنگامی که یک شئ مورد دسترسی اجرای آن زمان‌گیر است، برای نمونه محاسبات سنگینی را انجام می‌دهد و یا اینکه نیاز به ورودی/ خروجی زیادی از سند یا از طریق کاربر دارد، مسلماً، کار آزمون اشیایی که آن را مورد دسترسی قرار می‌دهند دچار وقفه و بعضاً بسیار زمان‌گیر و غیرممکن می‌گردد. لذا، در این‌گونه موارد استفاده از اشیا جاعل ضروری می‌باشد[۳].

در برخی از موارد، استفاده از آزمون واحد ضروری می‌باشد، به‌خصوص هنگامی که در مقایسه با شئ مورد آزمون اشیاء محیطی آن بسیار پیچیده‌اند. ممکن است در طی فرایند آزمون، اشیاء محیطی هنوز ایجاد نشده باشند و یا نتایج غیر قابل پیش‌بینی ایجاد کنند یا مقادیری ایجاد کنند که محاسبات بسیار زیادی نیاز دارند.

هنگام آزمون یکپارچگی، راهکارهای آزمون واحد مورد استفاده قرار می‌گیرد. بر اساس این آزمون، مشخص می‌شود آیا این قطعات بدان صورتی‌که مورد انتظار است با یکدیگر در تعامل قرار می‌گیرند [۱۲و۳].

۲-۳. شیء جاعل Deja-Vu

راه‌کار استفاده از اشیاء جاعل برای آزمون واحد اشیاء، مسائل و مشکلات جانبی خود را دارند. مشکل زمانی است که یک شئ جاعل حجم عظیمی از داده‌ها را به‌واسطه یک پرس و جو SQL برمی‌گرداند. می‌توان با آزمون پایین به بالا کار ایجاد اشیاء جاعل را تسریع نمود. برای این منظور با آزمون اشیاء مستقل در برگ‌های درخت وابستگی اشیاء مورد آزمون، کار تولید اشیاء جاعلی آسان می‌گردد.

در این راستا، راه‌کاری مبتنی بر استفاده از اشیایی به نام Deja-vu مطرح شده است. این نوع اشیاء به‌طور خودکار ایجاد می‌شوند، بنابراین کار دستی زیادی برای ایجاد آنها لازم نیست.

اشیاء Deja-Vu قادر به مشاهده و ثبت رفتار اشیاء واقعی هستند و در هنگام آزمون با آنها جایگزین می‌شوند و مقادیر ثبت شده را انتخاب و برگشت می‌دهند[۸]. لازم به ذکر است که شی جاعل Deja-Vu از جدول برای نگهداری اطلاعات استفاده می‌کند[۱۲و۳].

۳. روش پیشنهادی

همانطور که در قسمت‌های قبل توضیح داده شد، روش آزمون به صورت دستی دارای مشکلاتی می‌باشد؛ در این قسمت، به ارائه چند روش برای حل این مشکل خواهیم پرداخت. روش‌های ارائه شده به دو قسمت کلی تقسیم می‌شوند:

- تعیین درصد بهینه دادگان آزمون
- ایجاد شئ جاعل با کمک شبکه عصبی

۳-۱. روش پیشنهادی تعیین درصد بهینه دادگان آزمون

در این قسمت برای تولید خودکار داده‌ها از شبکه عصبی استفاده شده است. می‌توان میزان خطای شبکه عصبی را با تعیین درصد ورودی‌های مورد استفاده برای آموزش شبکه نسبت به کل نمونه‌های پایگاه داده کاهش داد. برای این منظور، با استفاده از الگوریتم‌های ژنتیک و الگوریتم ذوب فلز شبکه عصبی در چند مرحله آموزش داده می‌شود و به مرور در طی یک فرایند تکاملی، بهترین شبکه با کمترین میزان خطا مشخص می‌شود.

چارچوب کلی برای یافتن شبکه بهینه به صورت ذیل است. برای شبیه‌سازی عملکرد توابعی که از طریق فراخوانی شئ مورد آزمون به‌دست می‌آید و یا اینکه اشیا فراخوانی شده زمان اجرا طولانی دارند. وقت زیادی از شئ مورد آزمون صرف شئ فراخوانی می‌شود. با استفاده از شبکه‌های عصبی می‌توان تابع مورد نظر را چندین بار با داده‌هایی آزمایشی به اجرا در می‌آید و نتیجه حاصل از اجرای آن ثبت می‌شود.

همان‌گونه که در شکل(۳) مشخص شده است، با استفاده از مجموعه ورودی‌ها و خروجی‌های متناظر با آنها، شبکه عصبی در طی یک فرایند یادگیری ایجاد و مورد آزمایش و بررسی قرار می‌گیرد.

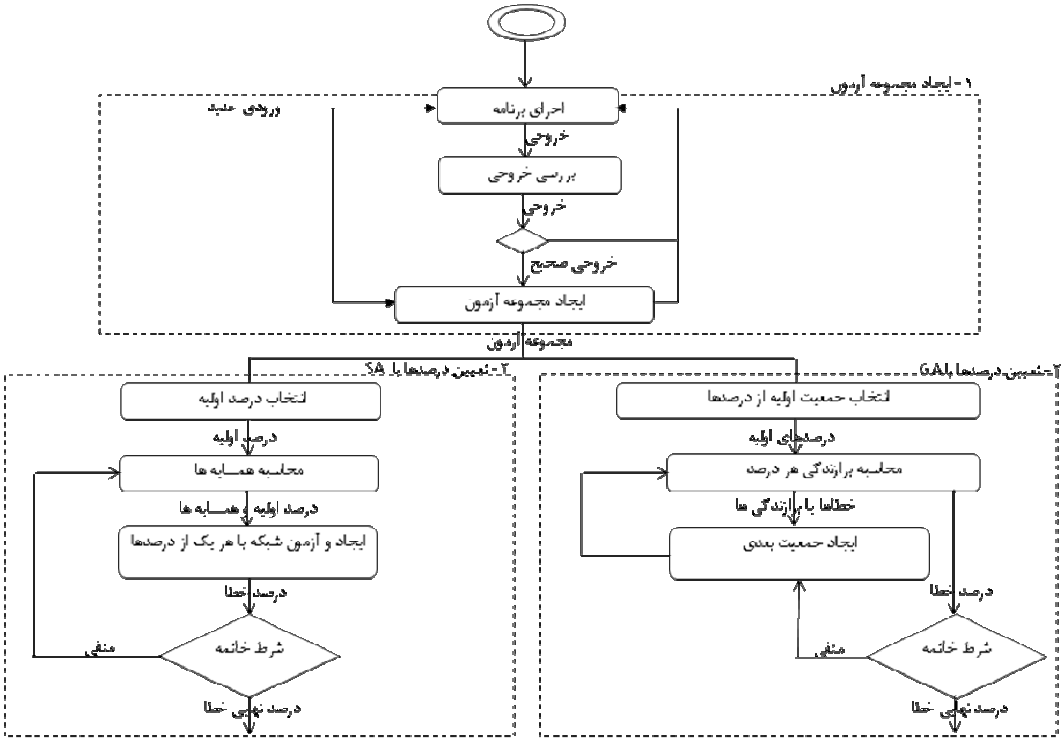
برای ایجاد شبکه عصبی، در ابتدا مجموعه‌ای از ورودی‌ها مشخص می‌شود. سعی می‌شود که تعداد ورودی‌ها به اندازه‌ای باشد که شبکه مورد نظر به صورت دقیق آموزش ببیند تا بتواند کلیه مسیرهای یک شئ را مورد آزمون قرار داد و از صحت درستی شئ اطمینان حاصل شود. نتایج حاصل از این ورودی‌ها با اجرای تابع مورد نظر مشخص می‌شود.

به این ترتیب مجموعه‌ای از ورودی‌ها و خروجی‌های متناظر با آنها مشخص می‌شود. به این مجموعه در اصطلاح مجموعه موردهای آزمون گفته می‌شود.

نکته حایز اهمیت تعیین تعدادی از این موردها از درون مجموعه برای آموزش و ایجاد شبکه است. برای تعیین این تعداد یا در واقع درصد از کل مجموعه داده شده یک بار از الگوریتم ژنتیک و بار دیگر از الگوریتم ذوب فلز استفاده شد.

۳-۲. روش پیشنهادی ایجاد شئ جاعل با شبکه عصبی

رفتار شئ ممکن است به صورت ایستا و یا پویا و در هنگام اجرای برنامه مورد بررسی قرار گیرد. رفتار در دنباله‌ای از فراخوانی‌های روش‌های شئ مشخص می‌شود. هنگامی که اجرای روش‌های یک شئ زمان‌گیر است، جایگزینی شئ زمان‌بر با شئ جاعل برای آزمون اشیایی که شئ را مورد دسترسی قرار می‌دهند، غیر قابل اجتناب می‌باشد. در واقع شئ جاعل با اجرای شئ واقعی ایجاد می‌شود. به این ترتیب که روش‌های شئ واقعی چندین بار به اجرا در می‌آید و در هر اجرا ورودی‌ها شامل نام روش، پارامترهای روش‌ها و ویژگی‌ها یا زمینه‌های شئ و خروجی حاصل از هر روش در ردیفی از یک جدول ذخیره می‌شوند. با استفاده از این جدول، شئ جاعل ایجاد می‌شود.



شکل ۳. فرآیند تولید داده خودکار و بهینه‌سازی آزمون شئ جاعل

مشکل در اینجا است که ورودی‌ها برای فراخوانی روش‌های شئ جاعل محدود به پارامترهای ارسالی است که در طی اجراهای آزمایشی شئ واقعی گردآوری شده است، در صورتی که ممکن است پارامترهای ورودی وابسته به شرایط اجرایی فراخواننده تغییر کنند. لذا، نیاز به روشی است که بتوان عملکرد شئ جاعل را نمونه‌سازی کند. در این راستا استفاده از مکانیزم‌های یادگیری ماشین مثل شبکه‌های عصبی می‌تواند بسیار مفید باشد[۱۳].

در این مقاله با استفاده از شبکه عصبی پرسپترون[1] چند لایه عملکرد شی مورد دسترسی را شبیه‌سازی کرده و یک شی جاعل ایجاد می‌کنیم. در واقع با ایجاد شبکه عصبی، نگاشتی بین ورودی و خروجی‌های برنامه مورد نظر ایجاد می‌شود. این نگاشت یک رابط ریاضی می‌باشد که عملکرد برنامه مورد نظر را شبیه‌سازی می‌نماید. بدین وسیله با مقایسه خروجی‌های برنامه با خروجی‌های پیش‌بینی شده توسط شبکه عصبی می‌توان صحت خروجی‌های برنامه را ارزیابی نمود. شبکه عصبی مطرح شده به دو فرم آموزش داده می‌شود، انتشار رو به عقب و الگوریتم ژنتیک که در شکل(۴) دیده می‌شود.

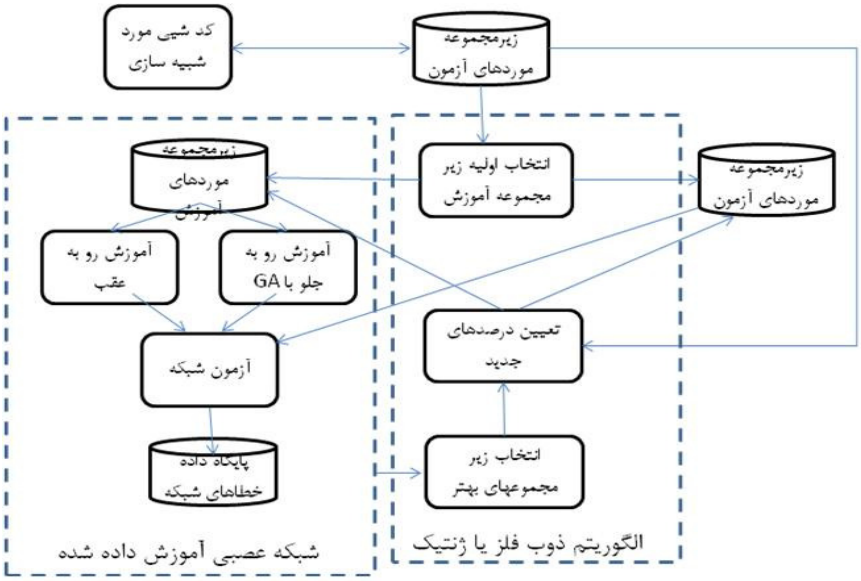
هدف از ارائه الگوریتم حاضر به حداقل رساندن میزان خطا در تخمین توابع توسط شبکه عصبی است. برای این منظور در ادامه دو الگوریتم ارائه شده برای این منظور از یک الگوریتم ژنتیک و یک الگوریتم ذوب فلز برای جستجو جهت یافتن وزن بهینه استفاده شده است. برای بهبود فرآیند یادگیری از دو روش انتشار رو به عقب و الگوریتم‌های ژنتیک برای یادگیری وزن‌های شبکه عصبی استفاده شده است.

[1] Multi-Layer Perceptron (MLP)

۴. نتایج و بحث

ابتدا نرخ بهینه داده‌های مرحله‌های آموزش و آزمون به کل داده‌ها را از طریق الگوریتم ژنتیک تخمین زده شده است. برای این منظور، از خطای یادگیری شبکه عصبی به‌عنوان برازندگی یا Fitness کروموزوم-ها استفاده شده است. هر کروموزوم یک عدد اعشاری می‌باشد که یک نرخ یا درصد را به عنوان نسبت داده‌های آموزش به کل داده‌های مشخص می‌کند. در ادامه سعی شده است با به‌کارگیری یک الگوریتم ذوب فلز به جای الگوریتم ژنتیک نرخ بهینه محاسبه شود.

شبکه عصبی، دارای ۳ لایه است که لایه میانی از ۵ نرون و لایه ورودی و خروجی، بستگی به نوع توابع دارد. برای مثال، تابع مرتب سازی دارای ۱۰ گره در لایه ورودی و ۱۰ گره در لایه خروجی می‌باشد. تنظیمات شبکه عصبی یعنی تعداد لایه‌ها و تعداد گره‌های لایه میانی برای تمام آزمایش‌ها ثابت می‌باشد.



شکل ۴. چارچوب تعیین آزمون و آموزش اشیاء

علت استفاده از این توابع بنیادی این است که کلیه توابع دیگر را با استفاده از این توابع می‌توان به راحتی پیاده‌سازی و صحت و درستی این توابع را بهتر بررسی کرد.

پس از تعیین درصد دادگان آزمون توسط الگوریتم فوق، شبکه عصبی آموزش داده می‌شود و نرخ خطای یادگیری محاسبه می‌شود. این عمل برای هر دادگان ده مرتبه اجرا شده و میانگین آن گزارش شده است.

به همین ترتیب، الگوریتم ذوب فلز برای جستجو جهت یافتن درصد بهینه موردهای آموزش و آزمون نسبت به کل موردها برای کلیه توابعی که به عنوان شئ جاعل جهت بررسی و آزمون اشیاء فراخوانی شده مورد استفاده قرار می‌گیرند. نتایج در جداول (۱) و (۲) آورده شده است. مقادیر جداول پس از ده مرتبه اجرا به صورت مستقل و میانگین‌گیری گزارش شده است. در ستون آخر جدول اختلاف خطا که از داده اولیه و داده‌هایی که از شبکه عصبی به دست آمده درج شده است.

جدول ۱. نتایج حاصل از الگوریتم ژنتیک

	داده‌های آزمون	داده‌های آموزش	خطا
Sum	46.38	53.62	0.0003
Sub	46.07	53.93	0.0001
Mul	48.22	51.78	0.4636
Sort	31.57	68.43	2.3054
Max	32.40	67.60	0.9793
Min	36.24	63.76	0.9885
AND	49.28	50.72	0.0000
OR	52.21	47.79	0.0000
AND_AND	52.04	47.40	0.1658
OR_OR	50.10	49.90	0.0000
AND_OR	49.77	50.23	0.2452

جدول ۲. نتایج حاصل از ذوب فلز

	خطا	داده‌های آموزش	داده‌های آزمون
Sum	0.0002	52.5	47.5
Sub	0.0001	68.7	31.3
Mul	0.5660	53.6	46.4
Sort	2.2286	82.4	17.5
Max	0.9651	81.7	18.3
Min	0.9328	86.3	13.7
AND	0.0000	68.1	31.9
OR	0.0000	44.9	55.1
AND_AND	0.0000	48.4	51.6
OR_OR	0.0000	64.3	35.7
AND_OR	0.0000	51.7	48.3

همچنین خطای گزارش شده به‌صورت میانگین مربعات خطا می‌باشد. همانطور که مشاهده می‌شود، خطای شبکه عصبی در تعیین درصد بهینه با روش شبیه‌ساز ذوب فلز نسبت به الگوریتم ژنتیک کمتر است اما درصد دادگان آموزش بسیار بیشتر می‌باشد.

لازم به ذکر است که بیشتر بودن دادگان آموزش باعث افزایش قدرت یادگیری سامانه می‌شود و ما دنبال این هستیم که حداقل درصد را تعیین کنیم، بدون آنکه خطا زیاد شود. در ادامه به بررسی روش پیشنهادی ایجاد شئ جاعل به کمک شبکه عصبی می‌پردازیم.

شبکه عصبی دارای سه لایه بوده که لایه میانی از ۵ نرون و لایه ورودی و خروجی بستگی به نوع توابع دارد. برای مثال، تابع مرتب سازی دارای ۱۰ گره در لایه ورودی و ۱۰ گره در لایه خروجی می‌باشد. تنظیمات شبکه عصبی یعنی تعداد لایه‌ها و تعداد گره‌های لایه میانی برای تمام آزمایش‌ها ثابت می‌باشد.

هر کدام از نتایج در جدول (۳) نشانگر میانگین خطای مقدار تولید شده توسط شبکه عصبی و مقدار واقعی بر روی ۱۰ بار اجرای مستقل می‌باشد. تفاوت مقدار واقعی با مقدار محاسبه شده، توسط شبکه عصبی ایجاد شده با الگوریتم ژنتیک نسبت به روش انتشار رو به عقب بسیار زیاد است. همچنین، در مورد سایر توابع در مجموع روش انتشار رو به عقب نتیجه بهتری را حاصل نموده است.

خطاهای مشخص شده در این محاسبات نرم، از اختلاف بین داده‌های مورد آزمون اولیه با داده‌هایی است که توسط محاسبات نرم محاسبه و با استفاده از این محاسبه، تخمین خطا برای آزمون نرم‌افزار شئ‌گرا محاسبه و تعین می‌شود(حدود اشتباه نرم‌افزار مشخص می‌شود). اگر نرم‌افزار با این مقدار خطا مشکلی ندارد، با بیش از این مقدار، نرم‌افزار دارای خطا می‌باشد. سپس مکان خطا را پیدا کرده و مشکل خطای نرم‌افزار شئ‌گرا رفع می‌شود.

جدول ۳. نتایج ۱۰ مرتبه اجرای الگوریتمNN_BP و NN_GA

	NN-BP	NN-GA
Sum	0.0266	500.5771
Sub	0.0177	252887.7
Mul	35.0934	507.8744
Sort	0.0006	13.1717
Max	1.1503	29.4957
Min	1.1371	27.9186
AND	0.0000	4.8859
OR	0.0000	4.0341
AND_AND	0.0000	4.9164
OR_OR	0.0000	5.2194
AND_OR	0.0000	6.6928

در جدول (۳) نتایج به دست آمده با کمک شبکه عصبی که توسط الگوریتم یادگیری بر اساس ژنتیک و یادگیری بر اساس روش انتشار رو به عقب آموزش داده شده است، ارائه شده است. همانطور که در جدول نشان داده شده است، الگوریتم انتشار رو به عقب همواره نتایج بهتری نسبت به الگوریتم ژنتیک تولید کرده است. بنابراین، می‌توان نتیجه گرفت الگوریتم انتشار رو به عقب در کاربرد مورد نظر می‌تواند به‌جای الگوریتم ژنتیک استفاده شود.

۵. نتیجه‌گیری

برای آزمون شئ، نیاز است روش‌هایی که با آن در ارتباط هستند، با ایجاد درخت فراخوانی شئ مورد آزمون مشخص شوند. سپس اشیاء مورد فراخوانی به‌صورت مستقل مورد بررسی و آزمون قرار می‌گیرند. با کمک شبیه‌سازی رفتار شئ، شئ جاعل ایجاد می‌شود. نگهداری مقدار پارامترهای شئ فراخوانی در یک جدول روشی است که دارای مشکلاتی است.

این روش نیاز به تعداد زیادی نمونه از مقادیر ممکن پارامترهای توابع دارد، همچنین دارای دقت پایین به علت عدم امکان ذخیره تمام مقادیر ممکن پارامترهای ورودی توابع و زمان‌بر بودن به علت نیاز به جستجو در جدول از مشکلات شی جاعل مبتنی بر جدول است.

در این مقاله استفاده از شبکه عصبی برای پیاده‌سازی شئ جاعل پیشنهاد شده است. مزایای راهکار پیشنهادی، یادگیری رابطه بین ورودی‌ها و خروجی‌های تابع با استفاده از تعداد محدودی از نمونه‌ها، دقت بالاتر در مقایسه با شئ جاعل مبتنی بر جدول و سریع بودن شبکه عصبی در تولید خروجی است. چالش‌های پیش روی روش پیشنهادی، یافتن بهترین وزن‌های ممکن برای نرون‌های شبکه عصبی می‌باشد.

از شبکه عصبی برای شبیه‌سازی عملکرد توابع بنیادی ریاضی، منطقی و گسسته استفاده شده است. برای یادگیری وزن‌های نرون‌های شبکه عصبی، از دو روش انتشار رو به عقب و الگوریتم ژنتیک استفاده شده است. همچنین از دو الگوریتم شبیه‌سازی ذوب فلز و ژنتیک برای تعیین درصد داده‌های آموزش و آزمون استفاده شد که در این میان، الگوریتم شبیه‌سازی ذوب فلز کارایی بهتری را در پی داشته است. پیاده‌سازی شئ جاعل به کمک شبکه عصبی، باعث افزایش سرعت آزمون در برنامه‌های شئ گرا شده است.

۶. مراجع

[1]. Binder, R. “Testing Object-Oriented Systems: Models, Patterns, and Tools.”; Reading Mass.: Addison-Wesley, 1999.

[2]. Jerry Gm, Z.; Kung, D. “Object State Testing for Object-Oriented Programs.”; IEEE Computer Software Application Conference, 1995.

[3]. Afsharian. S.; Bei, A.; Bianchi, M. “Unit Testing with Déjà-Vu Objects.”; World Academy of Science, Engineering and Technology 2006, 22.

[4]. Zhao. R.; Lv, S. “Neural-Network Based Test Cases Generation Using Genetic Algorithm.”; 13th IEEE International Symposium on Pacific Rim Dependable Computing, 2007.

[5]. Sun, T.; Chow, S. “Testing Software Design Modeled by Fiite-State Machines.”; IEEE TSE, 4(3), 1978, 178-187.

[6]. Edvardsson, J. “A Survey on Automatic Test Data Generation.”; In Proceedings of the Second Conference on Computer Science and Engineering in Linkoping, 1999, 21–28.

[7]. Sexton, R. S.; Dorsey, R. E.; Johnson, J .D. “Optimization of Neural Networks: a Comparative Analysis of the Genetic Algorithm and Simulated Annealing.”; European Journal of Operational Research 1999, 114, 589-601.

[8]. Corana, A.; Marchesi, M.; Martini, C.; Ridella, S.; “Minimizing Multimodal Functions of Continuous Variables with the Simulated Annealing Algorithm.”; ACM Transactions on Mathematical Software 1987, 13, 262-280.

[9]. Dorsey, R. E.; Mayer, W. J. “Optimization Using Genetic Algorithms.”; Advances in Artificial Intelligence in Economics, Finance, and Management, Greenwich, CT: JAI Press Inc. 1994, 1, 69-91

[10]. Meyer, B.; Ciupa, I.; Leitner, A.; Liu, L. “Automatic Testing of Object-Oriented Software.”; in Proceedings of SOFSEM 2007, Current Trends in Theory and Practice of Computer Science, 2007.

[11]. Brown, M. A.; Tapolcsanyi, E. “Mock Object Patters.”; in Proceedings of the 10th Conference on Pattern Languages of Programs (PLOP03), 2003.

[12]. Mackinnon, T.; Freeman, S.; Craig, P. “Endo-Testing: Unit Testing with Mock Objects.”; in Proceedings of the International Conference on Extreme Programming and Flexible Process in Software Engineering, (XP2000).

[13]. Archer, N.; Wang, S. “Application of the Back Propagation Neural Network Algorithm with Monotonicity Constraints for Two-Group Classification Problems.”; Decision Sciences, 1993, 24(1), 60-75.

[14]. Binder, R. “Testing Object-Oriented Systems: Models, Patterns, and Tools.”; Reading Mass.: Addison-Wesley, 1999.

[۱۵]. ترکاشون، رضا و کنگاوری، محمدرضا "ارائه شی جاعل مبتنی بر شبکه عصبی برای آزمون مستقل اشیاء"، نوزدهمین کنفرانس برق، دانشگاه امیرکبیر، ۱۳۹۰.