ششمین همایش بین‌المللی افق های نویـن در
مهندسی برق، کامپیوتـر و مکانیک
6th International Conference on the New Horizons in
Electrical Engineering, Computer and Mechanical
www.mhconf.ir

تولید کیس های آزمایشی مبتنی بر مدل با استفاده از الگوریتم های بهینه سازی ژنتیک و بهینه سازی تجمع ذرات

محمدجوادحسین پور*[۱]، احمدرضا ورزنده[۲]

*[۱] عضو هیات علمی و استادیار بخش مهندسی کامپیوتر، دانشگاه آزاد اسلامی واحد استهبان، استهبان، ایران

Email: hosseinpoor.mohammadjavad@gmail.com

[۲]دانشجوی کارشناسی ارشد، موسسه آموزش عالی اندیشه جهرم، جهرم، ایران

Email: varzandeh1988@gmail.com

چکیده : تست مبتنی بر مدل یکی از ابزارهای مهم جهت تولید خودکار موارد آزمایشی در نرم افزارهای میدانی و کاربردی است. اینکه یک نرم افزار بتواند کیفیت خدمت رسانی را افزایش داده و نیازهای کاربران را برآورده نماید؛ به صورت مستقیم به مواردی همچون پوشش پذیری خطا و غلبه بر پاسخ های بدون پشتوانه وابسته میباشد. در الگوهای نرم افزاری مبتنی بر عامل و نرم افزارهای سرویس گرا؛ رفتارهای سیستم در طول زمان تغییراتی زیادی نموده و وضعیت آن به صورت مرتب تغییر میکند. این مساله باعث میشود؛ درک همه جانبه این تغییرات با چالش های جدی روبرو گردیده و همواره یک فضای مساله بزرگ از رفتارهای گوناگون ماژول های تشکیل دهنده نرم افزارها پیش روی طراحان قرار گیرد. در این راستا در پژوهش جاری سعی شده است، ابتدا یک مدل از رفتارهای آنی نرم افزارها شامل سطح تحمل پذیری خطا و زمان اجرا، تولید گردیده و کیس های آزمایشی مختلف مورد بررسی قرار گیرند. بدیهی است؛ کیس های آزمایشی جهش یافته ای که بتوانند جنبه های تازه ای از عملکرد نرم افزاری را تعریف نمایند به عنوان ابزارهای جدید شناخته شده و به نسخه های آتی اضافه گردند.

کلید واژه‌ها: الگوریتم های فرااکتشافی، بهینه سازی، کیس های آزمایشی مبتنی بر مدل، یادگیری ماشین.

۱- مقدمه

آزمایش های دوره ای در پروژه های توسعه یافته بسیار مهم بوده و به موفقیت پروژه های فناوری اطلاعات کمک قابل توجهی میکند. کیفیت سیستم توسعه یافته و مطابقت آن با نیازهای مشتری از مسائلی است که توسط آزمایشهای نرم افزاری تضمین میگردد. جهت تصدیق عملکرد کامل آزمون، تمام الزامات مشخص شده باید توسط موارد تست مربوطه پوشش داده شوند[۱]. با توجه به افزایش سرعت توسعه در پروژه‌های فناوری اطلاعات، موارد آزمایشی باید فوراً ایجاد گردیده تا بازخورد فوری در مورد وضعیت فعلی سیستم دریافت شود. این امر مستلزم ارائه یک مدل واحد در ایجاد موارد آزمایشی است. به بیان دیگر، نمایش رفتار جاری سیستم توسط مدل های آزمایشی، مساله ای است که روند تغییرات یک نرم افزار در آینده را پیش بینی میکند[۲].

مدل‌های آزمایشی نه تنها به عنوان ابزاری برای تولید خودکار کیسهای آزمایشی عمل می‌کنند، بلکه امکان تشخیص زودهنگام خطاهای احتمالی در تطابق امکانات و نیازهای کاربران را نیز فراهم می آورند. علاوه بر این، آنها به تیم ها اجازه میدهند تا به طور شفاف در مورد طراحی سیستم مورد نظر بحث کنند. در گذشته، تولید نمونه‌های از مدل‌های آزمایشی به طور گسترده‌ای خودکار بوده است [۳]، در حالی که امروزه متخصصین هنگام ایجاد و نگهداری این مدل‌ها با مشکلات مختلفی مواجه می‌شوند!.

تولید مدلهای آزمایشی به چندین فاز اساسی تقسیم میگردد که در حال حاضر بیشتر به صورت دستی انجام میپذیرد. طراحان آزمون باید اطلاعات مربوطه را از مشخصات نیازمندیها که عمدتاً به زبان طبیعی (**NL**) نوشته شده است، شناسایی و استخراج نموده، آنها را به طور کامل درک کرده و در نهایت به یک مدل آزمون مناسب تبدیل نمایند. به ویژه در پروژه‌های نرم‌افزاری بزرگ، استخراج این الزامات چالش‌برانگیز است زیرا اسناد به سرعت پیچیده می‌شوند و علاوه بر نیازمندی‌های واقعی، شامل طیفی از اطلاعات دیگر (تعریف پارامترها، اطلاعات زمینه، مثال‌ها و غیره) نیز می‌شوند [۴]. علاوه بر این، پیچیدگی نیازمندی‌ها همزمان با افزایش پیچیدگی سیستم‌های فناوری اطلاعات افزایش می‌یابد چرا که نیازمند در نظر گرفتن تعدادی ورودی و خروجی مختلف هنگام ایجاد مدل‌های آزمایشی است. این مساله منجر به فرآیند ایجاد مدل آزمایشی زمان بر و مستعد خطا میشود زیرا کیفیت مدل آزمایشی به شدت به تواناییهای طراح آزمون بستگی دارد [۵]. عدم خودکارسازی فرآیند تولید کیس های آزمایشی نیز منجر به هزینه های اضافی در تعمیر و نگهداری نرم افزار می گردد. هر تغییری در یک نیاز، مستلزم بازنگری در مدل آزمایشی است. بنابراین؛ ایجاد یک ساختار اولیه از کیس های آزمایشی بالقوه و اولویت بندی آن ها؛ راهکاری است که اخیرا پیش روی محققین قرار گرفته است. در اولویت بندی کیس های آزمایشی دو سوال اساسی مطرح است که عبارتند از [۶]:

۱) – آیا کیس آزمایشی تولید شده، میتواند زمان اجرای سیستم را به چالش بکشد؟

۲)- آیا کیس آزمایشی تولید شده؛ خطاهایی را در زمان اجرا خواهد داشت؟ (خطا از عدم برآوردن نیازهای اساسی کاربران).

همانطور که مشخص است؛ مولفه های یک مدل مناسب؛ پاسخ دادن به سوالات فوق بوده و مدلهایی که بتوانند توازن میان زمان اجرا و همپوشانی خطا را با موفقیت انجام پشت سر بگذارند، به عنوان کیس های آزمایشی نهایی روی نرم افزار اجرا میگردند. حال مساله اساسی؛ تولید کیس های آزمایشی از میان هزاران ترکیب مختلف است که یک فضای مساله بزرگ را پیش روی محققین قرار میدهد. به همین منظور، ابتدا یک مجموعه توانی از کیس های آزمایشی تولید گردیده و سپس توسط روشهای مبتنی بر ساختارهای سلسله مراتبی؛ نمونه های موثر

انتخاب میگردند [۷]. اینکه نحوه پویش فضای مساله ی ساخته شده از کیس های آزمایشی به چه صورت باشد، چالشی است که توسط الگوریتمهای بهینه سازی گسسته به آن پاسخ داده میشود (چالش اصلی پژوهش جاری).

در ادامه این مقاله، ابتدا به شرح جزئیات روش پیشنهادی پرداخته شده و سپس الگوریتمهای مختلف تولید کیسهای آزمایشی مبتنی بر مدل با یکدیگر مقایسه شده اند. در انتها نیز نتیجه گیری نهایی از مزایا و معایب روش ارائه شده محقق مطرح شده است.

۲- متدولوژی

روند تولید کیسهای آزمایشی با توجه به تعداد بالای پاسخهای کاندید و همپوشانی برخی از آنها با یکدیگر، معمولا با پیچیدگی های خاصی همراه گردیده و مرتبه زمانی زیادی را طلب می نماید. به همین منظور؛ از تستهای مبتنی بر رگرسیون[1] استفاده میشود. در تستهای رگرسیونی، اطمینان حاصل میگردد که تغییرات در یک ماژولِ نرم افزار بر روند کارِ دیگر ماژول ها تاثیر معکوس نمی گذارد! به بیان دیگر، تست رگرسیونی، پروسه ای است که به نگهداری سیستم های نرم افزاری پس از ارائه آنها به مشتری اشاره دارد [۸].

در این گونه تستها، رفتارهای نرم افزار هدف به صورت کیس های آزمایشی ارائه گردیده و خطاهای تشخیص داده شده در واحد زمان، عملکرد مدل را ارزیابی میکنند. تعداد بالای کیسهای آزمایشی باعث میشود الگوریتم های اجرای تست های رگرسیونی، مرتبه زمانی نمایی را تجربه نماید [۹]. به همین منظور، جهت غلبه بر پیچیدگی های موجود، انتخاب کیس های آزمایشی توسط الگوریتم های فرااکتشافی انجام میپذیرد. الگوریتم های فرااکتشافی؛ با پویش وسیع فضای مساله؛ ترکیباتی را انتخاب میکنند که نسبت به سایرین از خروجی بهینه تری برخوردار هستند. بهینگی، توسط پارامترهای میزان خطای قابل پوشش و زمان اجرا تعریف میشود که تلفیق آنها با یکدیگر، منجر به ارائه یک تابع تناسب هدفمند میگردد.

در الگوریتم های فرااکتشافی؛ مساله بسیار مهم؛ هوشمندی عاملها در حرکت در فضای مساله است. در تمامی الگوریتم های حاضر در این حوزه، عاملها به صورت گروهی با یکدیگر در ارتباط بوده و به صورت غیر مستقیم بر روند حرکت دیگر عاملها تاثیر میگذارند. در این الگوریتم ها، مساله بسیار مهم، چگونگی برخورد عامل ها با نقاط بهینه و سرعت همگرایی آنهاست. در صورتیکه یک عامل به نقطه غیربهینه رسیده و دیگر عامل ها بدون درک درست از فضای مساله آن را دنبال کنند، پدیده بهینه محلی رخ میدهد که خروجی آن عدم انتخاب کیس های آزمایشی موثر در فضای مساله خواهد بود.

به همین منظور، در پژوهش جاری روشی مطرح شده است که میتواند توازن میان پاسخ های بهینه سراسری و محلی را برقرار نموده و کیفیت پاسخ های نهایی را تا حد زیادی بهبود بخشد. پیش از ارائه توضیحات در رابطه با روش پیشنهادی، لازم است تابع تناسب مرتبط با کیس های آزمایشی تولید شده توسط تست های رگرسیونی تشریح گردد.

۲-۱- تابع تناسب

[1] Regression Testing

تابع تناسب، بعنوان عملگر ارزیابی پاسخ های بهینه سازی شده مورد استفاده عامل ها قرار میگیرد. هر عامل؛ پس از آنکه کیس آزمایشی ای را از فضای مساله انتخاب نمود (فضای مساله تولید توسط تست رگرسیونی)؛ می بایستی از کیفیت پاسخ دهی آن مطلع شود. میزان تاثیرگذاری کیس های آزمایشی به صورت مستقیم به دو پارامتر زمان اجرا و قابلیت پوشش دهی خطاهای رخ داده مرتبط است که تلفیق آنها با یکدیگر، تشکیل تابع تناسب را میدهد. تعریف دقیق این دو پارامتر به شرح زیر است [۱۰]:

- **تعداد خطاهای تشخیص داده شده (*F*)**: در این پارامتر؛ مجموعه T = {T1,T2,T3,...,TN} شامل کیس های آزمایشی و F = {F1,F2,F3,...,FN} شامل خطاهای پوشش داده شده توسط هر کیس آزمایشی در واحد زمان میباشد (تابع F(T(i)) زیر مجموعه ای از خطاهای پوشش داده شده را تعیین مینماید). نحوه محاسبه این پارامتر در رابطه شماره ۱ نشان داده شده است.

$$Fault\ coverage = \frac{\sum_{t=1}^{S} U_{t=1}^{S}[F(Ti)]}{K} \qquad (۱)$$

- **زمان اجرای کیس های آزمایشی(*T*)**: این پارامتر؛ زمان اجرای کیس های آزمایشی را به صورت خطی و به کمک یک تابع سیگموید محاسبه مینماید.

نکته حائز اهمیت در بخش های مختلف تابع تناسب موجود، بیشینه سازی پارامتر تعداد خطاهای تشخیص داده شده و کمینه سازی زمان اجرای کیس های آزمایشی است که تلفیق آنها توسط رابطه شماره (۲) ارائه میگردد.

$$Fitness = \beta \frac{1}{F} + T \qquad (۲)$$

در رابطه شماره ۲، پارامتر ß یک ضریب تبادل (توازن) میان مولفه های زمان و تعداد خطاهای تشخیص داده شده می باشد که در این پژوهش؛ به آن مقدار ۰,۱ نسبت داده شده است (به این دلیل که متغیر تعداد خطاهای تشخیص داده شده معمولا چندین برابر متغیر زمان اجراست؛ تقسیم زمان بر تعداد خطاهای تشخیص داده شده؛ پارامتر ß را محاسبه مینماید). در نهایت می بایستی اضافه نمود که کمینه سازی رابطه شماره ۲ هدف اصلی تحقیق جاری می باشد.

۲-۲- الگوریتم پیشنهادی

با توجه به اهمیت بالای پویش وسیع فضای مساله و فرار از بهینه محلی در الگوریتم های بهینه سازی؛ در روش پیشنهادی محقق؛ الگوریتم های بهینه سازی تجمع ذرات و ژنتیک با یکدیگر ترکیب گردیده اند. دلایل ترکیب این دو الگوریتم به شرح زیر است:

۱. الگوریتم بهینه سازی تجمع ذرات از قدرت مناسبی در پویش وسیع فضای مساله[۲] و انتخاب بهینه سراسری برخوردار است.
۲. الگوریتم ژنتیک قدرت جستجوی محلی[۳] بهتری نسبت به سایر روش های بهینه سازی دارد.

[2] Exploration
[3] Exploitation

البته می بایستی توجه داشت روش های فوق معایبی نیز دارند که به شرح زیر است:

۱. الگوریتم بهینه سازی تجمع ذرات به صورت ذاتی دچار مشکل بهینه محلی است که این خصیصه به واسطه حرکت ناگهانی ذرات به سمت ذره بهینه یا G_{best} رخ میدهد.
۲. الگوریتم ژنتیک، به دلیل ترکیب کروموزوم های برتر در نسل های متوالی، نیاز به حافظه و زمان زیادی دارد که این مساله باعث میشود تضمینی در ارائه پاسخ بهینه سراسری در این روش وجود نداشته باشد!

با کمی تامل در مزایا و معایب روشهای فوق، میتوان دریافت که الگوریتم ژنتیک توانایی غلبه بر بهینه محلی را دارا بوده اما نیاز به حافظه بالایی جهت رسیدن به بهینه سراسری دارد. در طرف مقابل؛ الگوریتم بهینه سازی تجمع ذرات؛ با در نظر گرفتن ذره بهینه در هر دور؛ امکان انتخاب بهینه سراسری را فراهم میسازد اما به شرطی که دچار بهینه محلی نشود! بنابراین؛ روش هیبریدی محقق به شرح زیر خواهد بود:

۱. تولید فضای مساله از ترکیبات ممکن کیس های آزمایشی.
۲. تولید ذرات در فضای مساله به صورت تصادفی.
۳. ارزیابی قدرت پاسخ های ارائه شده توسط هر عامل (با استفاده از رابطه شماره ۲).
۴. محاسبه مقدار X_{best} در میان ذرات (ذره با کمینه ترین پاسخ).
۵. انتخاب دو ذره با تابع تناسب بهینه نسبت به سایرین به صورت تصادفی به عنوان ذرات والد (Parent1 , Parent2) (این پروسه به ازای تمامی ذرات اجرا میشود).
۶. حذف دو ذره با خروجی غیر بهینه از فضای مساله.
۷. تولید دو ذره جدید با ترکیب عرضی شماره کیس های آزمایشی[4] (رابطه ۳).

(۳)

$$\text{Particle 1} = \alpha \times \text{Parent1} + (1 - \alpha) \times \text{Parent2}$$

$$\text{Particle 2} = (1 - \alpha) \times \text{Parent1} + \alpha \times \text{Parent2}$$

(در رابطه ۳، منظور از α فاکتور یا ضریب وزن دهی می باشد که یک عدد تصادفی نرمال در بازه صفر و یک است).

۸. اعمال عملگر جهش با ضریب ۰٫۲[5] بر روی بردار پاسخ ذرات ساخته شده.
۹. به روز رسانی مقادیر X_{Best} و I_{Best} ذرات.

[4] Cross Over
[5] Mutation

۱۰. بازگشت به مرحله سوم و بررسی شرط پایان پذیری (همگرایی عامل ها).

۱۱. انتخاب بهترین پاسخ ارائه شده (ذره با برچسب G_{best}).

۱۲. انتخاب کیس آزمایشی موثر و تولید نمونه های مشابه.

با این ترکیب، میتوان امیدوار بود فضای مساله به صورت گسترده پویش گردیده و احتمال گرفتار شدن در بهینه محلی، به کمترین میزان خود برسد. یکی از مزایای مهم روش پیشنهادی محقق، ایجاد توازن[6] میان بهینه محلی و سراسری به واسطه جهش ذرات و جلوگیری از پرش آنی و غیرمنتظره آنهاست. این روند؛ به الگوریتم اجازه میدهد، ترکیبات جدیدی از کیس های آزمایشی را انتخاب نموده و تمرکز بیشتری بر انتخاب نقاطه بهینه سراسری داشته باشد.

۳- پیاده سازی و مقایسه

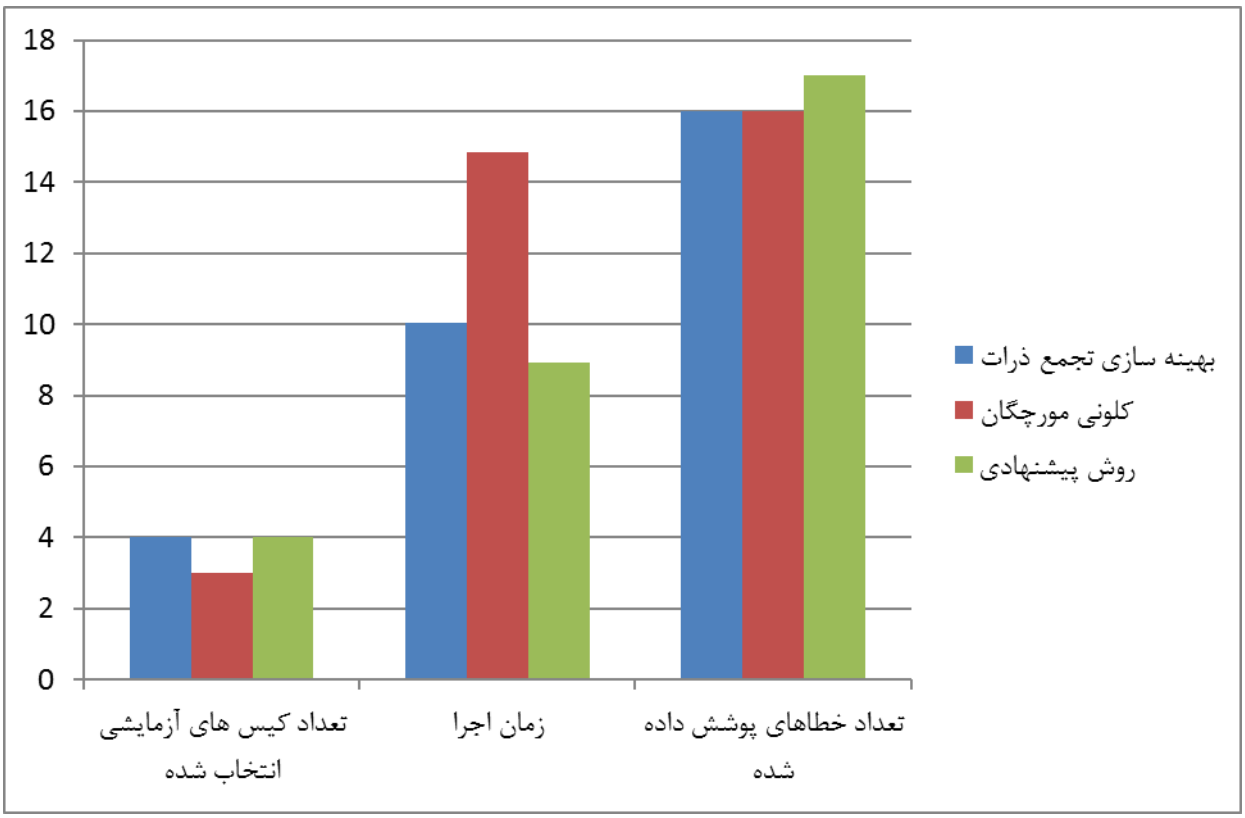
در[۱۰] الگوریتم های بهینه سازی تجمع ذرات و کلونی مورچگان بر روی مجموعه ای متشکل از ۵۶۷ کیس آزمایشی (و ۱۹ خطای سیستمی) تولید شده توسط آنالیزور لغوی Flex_Fast [۱۱] اجرا گردیده و نتایج آن در جدول شماره ۱ درج شده است. بنابر نتایج به دست آمده؛ میتوان اینگونه استنباط کرد که الگوریتم بهینه سازی تجمع ذارت نسبت به الگوریتم کلونی مورچگان، عملکرد مطلوب تری را ارائه نموده است که دلیل آن؛ حرکت عامل ها به سمت بهینه سراسری یا G_{best} می باشد. این در حالی است که روش پیشنهادی محقق در مقایسه با روش های فوق، عملکرد مطلوب تری را به نمایش گذاشته و کیفیت پاسخ های نهایی را بهبود بخشیده است (محیط پیاده سازی، زبان برنامه نویسی متلب نسخه ۲۰۱۹ در سیستمی با قدرت پردازشی ۲,۷ گیگاهرتز و ۵ هسته پردازشی بوده است).

جدول شماره ۱. نتایج به دست آمده از روش های بهینه سازی انتخاب کیس های آزمایشی موثر

نام الگوریتم	تعداد کیس های آزمایشی انتخاب شده	زمان اجرا	تعداد خطاهای پوشش داده شده
بهینه سازی تجمع ذرات	۴	۱۰,۰۶	۱۶

[6] Trade off

کلونی مورچگان	۳	۱۴,۸۶	۱۶
روش پیشنهادی	۴	۸,۹۴	۱۷



شکل ۱. نتایج به دست آمده از الگوریتم پیشنهادی محقق و دیگر روش های انتخاب کیس های آزمایشی

۴- نتیجه گیری

بررسی نتایج به دست آمده؛ نشان از برتری روش محقق در انتخاب کیس های آزمایشی موثر دارد. مهمترین مزیت روش پیشنهادی؛ ایجاد توازن میان پاسخ های بهینه محلی و سراسری است که به صورت مستقیم بر سرعت همگرایی عامل ها تاثیرگذار است. همچین؛ عدم محاسبه

چندین باره سرعت به ازای هر عامل و تولید تصادفی عامل های جدید از عامل های موثر دور فعلی؛ منجر به بهبود روند الگوریتم میشود. البته می بایستی به این نکته نیز اشاره داشت که روش پیشنهادی محقق؛ پس از ترکیب ذرات بهینه با یکدیگر و تولید ذرات جدید، به کمک اپراتور جهش؛ بخشی از بردار پاسخ را تغییر میدهد که این روند به بررسی ترکیبات بیشتر از فضای مساله کمک شایان توجهی خواهد نمود. همچنین به منظور پیشنهادهای آتی، میتوان از اپراتورهای ایجاد بی‌نظمی[۷] در پروسه تولید اعداد تصادفی استفاده نموده و هربار، عامل ها را در فضای مساله توزیع نمود. این کار؛ احتمال گرفتار شدن در بهینه محلی را به شدت کاهش میدهد اما میتواند زمان اجرای الگوریتم را نسبت به حالت اولیه آن افزایش دهد که این خصیصه با افزایش روز افزون قدرت دستگاه های پردازشی تقریبا قابل چشم پوشی است.

مراجع

[1] M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner, Model-Based Testing of Reactive Systems: Advanced Lectures, 2005.

[2] M. Utting, A. Pretschner, and B. Legeard, "A taxonomy of model-based testing approaches," Software Testing, Verification and Reliability, vol. 22,no. 5, 2012.

[3] Kramer, A. and Legeard, B., and Binder, R. V., "2016/2017 model-based testing user survey: Results," 2017. [Online]. Available: http://www.cftl.fr/wp-content/uploads/2017/02/2016-MBT-User-Survey-Results.pdf

[7] chaotic operator

[4] M. Kassab, C. Neill, and P. Laplante, “State of practice in requirements engineering: contemporary data,” Innovations in Systems and Software Engineering, vol. 10, no. 4, 2014.

[5] D. Freudenstein, J. Radduenz, M. Junker, S. Eder, and B. Hauptmann, “Automated test-design from requirements: The specmate tool,” in RET,2018.

[6] Shin, Ki-Wook, and Dong-Jin Lim. "Model-based automatic test case generation for automotive embedded software testing." *International Journal of Automotive Technology* 19.1 (2018): 107-119.

[7] Shah, Syed Asad Ali, et al. "Test case generation using unified modeling language." *2019 International Conference on Computer and Information Sciences (ICCIS)*. IEEE, 2019.

[8] Xu, Yiqun, and Linbo Wu. "An automatic test case generation method based on sysml activity diagram." *IOP Conference Series: Materials Science and Engineering*. Vol. 563. No. 5. IOP Publishing, 2019.

[9] Khatibsyarbini, Muhammad, et al. "Test case prioritization approaches in regression testing: A systematic literature review." *Information and Software Technology* 93 (2018): 74-93.

[10] Agrawal, Arun Prakash, and Arvinder Kaur. "A comprehensive comparison of ant colony and hybrid particle swarm optimization algorithms through test case selection." *Data engineering and intelligent computing*. Springer, Singapore, 2018. 397-405.

[11] Do, H., Elbaum, S., Rothermel, G.: Supporting controlled experimentation with testing techniques: an infrastructure and its potential impact. Empir. Softw. Eng. 10(4), 405–435 (2005).