مجلـه‌ی علمی پژوهش در علوم رایانه ، شماره ۱۴، تابستان ۱۳۹۸، ص ۴۹- ۳۳

ISSN: 2745-3673
http://www.Rcsj.ir

مجله‌ی علمی پژوهش در علوم رایانه (سال چهارم)
شماره ۱۴ / تابستان ۱۳۹۸ / ص ۴۹- ۳۳

مطالعه مروری بر تولید نمونه‌آزمون کمینه در آزمون تعاملی

سجاد اسفندیاری [۱]، وحید رافع [۲]

[۱] دانشجوی دکتری، دانشکده فنی مهندسی، دانشگاه اراک، اراک، ایران.

[۲] دانشیار، دانشکده فنی مهندسی، دانشگاه اراک، اراک، ایران.

نام نویسنده مسئول:
سجاد اسفندیاری

چکیده

آزمون نرم‌افزار یک فاز حیاتی درتوسعه نرم‌افزار است. اغلب، تعداد نمونه‌های آزمون بسیار زیاد است که بررسی تمامی آنها زمان‌بر می‌باشد. با حذف نمونه‌های آزمون افزونه می‌توان به زیر مجموعه‌ای از دنباله‌های آزمون رسید که پوشش کامل را در بر بگیرد. آزمون‌گر به مجموعه‌ای از، نمونه آزمون نیاز دارد که اهداف آزمون را ارضاء کند. این مجموعه از نمونه‌های آزمون به عنوان دنباله آزمون شناخته می‌شود. در هنگام تولید نمونه آزمون ممکن است نمونه آزمونی حذف شود که باعث ارضاء نشدن هدف آزمون گردد. بنابراین دنباله آزمون باید زیرمجموعه‌ای از دنباله‌های آزمون اصلی باشد که هدف آزمون را ارضاء کند. آزمون تعاملی معیارهای مختلفی برای تعیین هدف آزمون دارد که اصلی‌ترین آن آرایه پوشش است تکنیک‌های مختلفی برای تولید آرایه پوشش وجود دارد که به‌طور کلی به دو دسته روش‌های ریاضی و محاسباتی تقسیم می‌شوند روش‌های محاسباتی عمدتاً از استراتژی‌های محاسبات محض یا مبتنی بر هوش مصنوعی برای تولید آرایه پوشش کمینه در فضای جستجو استفاده می‌کنند. اهمیت آزمون تعاملی ما را برآن داشت که یک مطالعه مروری از کارهای صورت گرفته در این حوزه ارائه دهیم. یافته‌های حاصل از این مرور نشان می‌دهد استراتژی‌های مبتنی بر هوش مصنوعی قوی‌تر از استراتژی‌های مبتنی بر محاسبات عمل می‌کنند اما سرعت مناسبی ندارند.

واژگان کلیدی: آزمون نرم افزار، پوشش آرایه، آزمون تعاملی.

مجلـه‌ی علمی پژوهش در علوم رایانه ، شماره ۱۴، تابستان ۱۳۹۸، ص ۴۹- ۳۳

ISSN: 2745-3673
http://www.Rcsj.ir

مقدمه

با توجه به افزایش چشم‌گیر نیاز به محصولات نرم‌افزاری در جامعه کنونی، فرآیند آزمون نرم‌افزار از لحاظ کیفی و قابلیت اطمینان بسیار حائز اهمیت است. آزمون نرم‌افزار با هدف کشف خطاها و تضمین انطباق محصول با نیازمندی‌ها و انتظارات کاربر صورت می‌پذیرد. آزمون نرم‌افزار اغلب به دو دسته کلی آزمون جعبه سیاه (عملکردی) و آزمون جعبه سفید (ساختاری) تقسیم می‌شوند. در آزمون جعبه سیاه، آزمون‌کننده تنها مجموعه ورودی‌ها و خروجی‌های مورد انتظار را می‌شناسد. این آزمون مبتنی بر مشخصات نیازمندی‌ها است و نیازی به اجرای کد در آن نیست اما در آزمون جعبه سفید، توسعه دهنده نرم افزار می‌تواند به کد، طراحی و منابع دیگر نرم افزار دسترسی داشته باشد. در این حالت سیستم، آزمون را می‌توان به یک جعبه شیشه‌ای (جعبه سفید) تشبیه کرد که محتوای داخلی و نحوه عملکرد آن قابل رویت است. مشکل فرآیند آزمون نرم‌افزار، تعداد نمونه آزمون بسیار زیاد آن است که بررسی تمامی آن‌ها زمان‌بر است. با حذف نمونه آزمون افزونه می‌توان به زیرمجموعه‌ای از دنباله‌آزمون رسید که پوشش کامل را دربر بگیرد. آزمون‌گر به مجموعه‌ای از نمونه آزمون نیاز دارد که اهداف آزمون را ارضاء کند. این مجموعه از نمونه‌های آزمون به عنوان دنباله آزمون شناخته می‌شود. در هنگام تولید نمونه آزمون ممکن است نمونه آزمونی حذف شود که باعث ارضاء نشدن هدف آزمون گردد. بنابراین دنباله آزمون تولید شده باید زیر مجموعه‌ای از دنباله‌های آزمون اصلی باشد که هدف آزمون را براساس یک معیار خاص، ارضاء کند [1].

یکی از معیارهای کاهش نمونه آزمون، آزمون تعاملی t-تایی است که براساس ساختار آرایه پوشش (CA) یا آرایه متعامد (OA) دنباله آزمون کمینه را تولید می‌کند. t-تایی یک رویکرد ترکیبی است که در آن t نشان دهنده قوه تعامل است [2] و زیرمجموعه‌ای از پارامترهای ورودی به‌حساب می‌آید. پوشش آرایه CA (N; t, p, v) یک آرایه N*p است که هر زیر آرایه N*t تمام حالات ممکن از v نماد را حداقل یک‌بار پوشش می‌دهد. در این تعریف مقدار t ثابت است و آن را آرایه پوشش با قوه‌ی ثابت می‌نامند. اما اگر t تغییر کند آن را آرایه پوشش با قوه متغییر (VSCA) می‌نامند [3]. راهکارهای متعددی برای تولید دنباله آزمون پوشش آرایه با قوه ثابت و قوه متغییر ارائه شده است که عموما به دو دسته مبتنی بر محاسبات محض و مبتنی بر هوش مصنوعی تقسیم می‌شوند. [4].

ارزیابی استراتژی‌های موجود براساس دو پارامتر زمان تولید و اندازه آرایه دنباله آزمون صورت می‌گیرد [2]. استراتژی‌های مبتنی بر محاسبات محض معمولا دارای سرعت بسیار بالا و دقت پایین (اندازه آرایه بزرگ) هستند که در این بین استراتژی IPOG-D [5] بالاترین سرعت را دارد و استراتژی ITCH[6] دنباله آزمون را با کم‌ترین سایز آرایه استخراج می‌کند. از دیگر معیارهای ارزیابی می‌توان به اندازه t اشاره کرد که هرچه بیشتر باشد قدرت استراتژی بالاتر است. در این دسته قوی‌ترین استراتژی GTway [7] است که تا t = ۱۲ قادر به تولید دنباله آزمون است و ضعیف‌ترین آن استراتژی ITCH است که تا t = ۴ توان تولید دنباله آزمون را دارد. استراتژی‌های مبتنی برهوش مصنوعی به دلیل داشتن ساختار پیچیده دارای سرعت پایین و دقت بالایی هستند. این استراتژی‌ها در ابتدا حداکثر برای $t \leq 3$ دنباله آزمون تولید می‌کردند . [8] در این بین استراتژی SA بهترین دنباله آزمون را از لحاظ اندازه آرایه تولید می‌کند. بعدها این استراتژی‌ها توسعه یافتند و قوه‌های بالاتری را پشتیبانی کردند. استراتژی‌های CS [2]،PSTG [3]، GS [1] بهترین نتایج را برای $3 \leq t \leq 6$ استخراج می‌کنند. استراتژی GS تا t = ۲۰ دنباله آزمون تولید می‌کند و CS، سریع‌ترین استراتژی مبتنی بر هوش مصنوعی است.

در این مقاله مطالعه‌ای مروری بر کارهای صورت گرفته در حوزه تولید دنباله آزمون کمینه با پوشش آرایه با قوه ثابت و متغیر ارائه می‌شود و سعی می‌کنیم دسته‌بندی مناسبی از استراتژی‌های موجود را بر اساس سه پارامتر زمان، اندازه آرایه و قوه تعامل ارائه دهیم و نقاط ضعف و قوت هریک را بیان کنیم هدف اصلی از نگارش این مقاله آن است که پس از بررسی نقاط ضعف و قوت راهکارهای موجود به این سوال پاسخ داده شود که آیا امکان ارائه راهکاری که قادر باشد دنباله آزمونی را تولید کند که از هر سه لحاظ زمان، اندازه آرایه و قوه تعامل بهینه باشد، وجود دارد؟.

ادامه این مقاله به شرح ذیل می‌باشد: بخش ۲ مربوط به مفاهیم بنیادی در تولید دنباله آزمون است. بخش ۳ به بررسی استراتژی‌های تولید دنباله آزمون می‌پردازد. بخش ۴ شامل بحث و مقایسه استراتژی تولید دنباله آزمون است در این بخش استراتژی‌های مبتنی بر هوش مصنوعی و محاسبات محض را مقایسه نموده و نقاط ضعف و وقت هریک مورد بررسی قرار می-گیرد. بخش ۵ به نتیجه‌گیری اختصاص داده شده است.

مفاهیم بنیادی در تولید دنباله آزمون

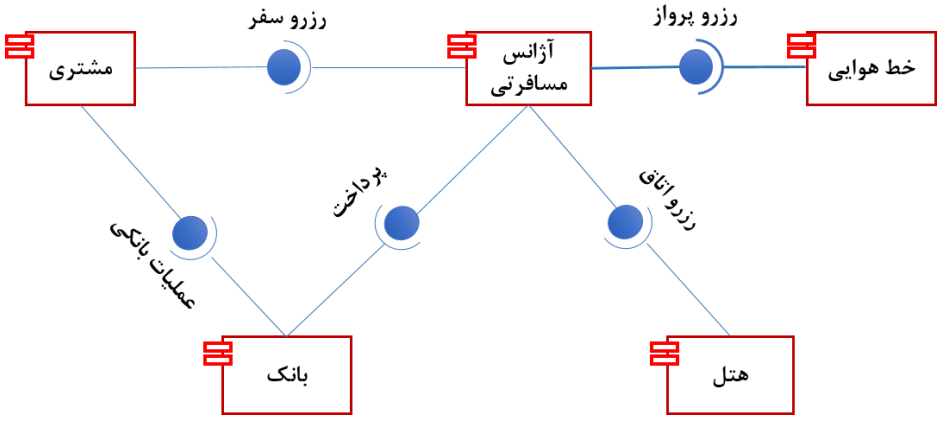
تعداد پارامترهای نرم‌افزار تحت آزمون معمولا بزرگ هستند و در صورتی که مقادیر این پارامترها هم بزرگ باشد، عملا بررسی تمام حالت-های (نمونه آزمون) آن ممکن نیست. یک روش ریاضی برای کاهش تعداد نمونه آزمون، پوشش آرایه است [2]. در آزمون ترکیبی، آرایه پوشش در ساده‌ترین شکل از یک جدول تشکیل می‌شود. هر سطر از این جدول معادل یک نمونه آزمون و ستون‌های آن، پارامترهای نرم‌افزار تحت آزمون است [9] و به عنوان ورودی آزمون ترکیبی محسوب می‌شوند. به طور کلی CA چهار پارامتر دارد: N، t، p و v که به دو شکل (۱) $CA(N,t,p,v)$ و (۲) $CA(N,t,v^p)$ نشان داده می‌شود. در این رابطه p تعداد پارامترها (یا فاکتورها)ی ورودی است، v تعداد مقادیری (یا level)

مجلـه‌ی علمی پژوهش در علوم رایانه ، شماره ۱۴، تابستان ۱۳۹۸، ص ۴۹- ۳۳

ISSN: 2745-3673
http://www.Rcsj.ir

است که هر پارامتر به خود اختصاص می‌دهد و t قوه تعامل (یا تعامل t-way) است. در واقع t زیرمجموعه‌ای از p است که به‌جای بررسی پوشش p پارامتر (ستون)، پوشش ترکیبات t از p بررسی می‌شود این حالت را پوشش آرایه با قوه ثابت نیز می‌گویند [3].

برای تشریح پوشش آرایه از مثال آژانس مسافرتی [10] به عنوان یک مثال اجرایی استفاده می‌کنیم. این سیستم از پنج جزء که عبارت است از: مشتری، آژانس مسافرتی، بانک، خط هوایی و هتل تشکیل شده است. ارتباط بین اجزا در شکل۱ نشان داده شده است. هر کدام از این اجزاء به عنوان یک پارامتر مستقل شناخته می‌شود و می‌توانند یک یا چند مقدار را بگیرند. جدول۱ این پارامترها و مقادیرشان را نشان می‌دهد. پنج‌تایی (وحید، سفیران، کاسپین، گلستان، سپه) یک نمونه آزمون از سیستم را تشکیل می‌دهد.



شکل ۱: اجزاء آژانس مسافرتی

جدول ۱: مقادیر اجزای آژانس مسافرتی

مشتری	آژانس مسافرتی	خط هوایی	هتل	بانک
وحید (۰)	رخش (۰)	زاگرس (۰)	پارسیان (۰)	سپه (۰)
سجاد (۱)	سفیران (۱)	کاسپین (۱)	گلستان (۱)	پاسارگاد (۱)
		آسمان (۲)		ملت (۲)

برای شناسایی خطا یا همان آزمون نرم‌افزار باید تمام تنظیمات متفاوت از نرم‌افزار را تولید کنیم. همان طور که گفتیم هر تنظیمات خاص از نرم‌افزار توسط یک نمونه‌آزمون مشخص می‌شود. مجموعه‌ای از این نمونه‌آزمون‌ها یک دنباله آزمون را تشکیل می‌دهند. برای شناسایی خطا باید دنباله آزمون ما تمام نمونه آزمون های ممکن را شامل شود.

برای مثال فرض خواهیم کرد که "سجاد" (مشتری (۱)) اجازه عملیات بانکی با بانک "سپه" را ندارد. پس زمانی که پارامتر مشتری، "سجاد" است و بانک "سپه" است خطای سیستم رخ می‌دهد. برای شناسایی این خطا حالت دوتایی (سجاد, -,-,-, سپه) حداقل ۱ بار باید توسط دنباله آزمون پوشش داده شود. این حالت را می توان به شکل (۱, -,-,-, ۰) نماش داد.

در آزمون کامل نرم‌افزار تمام p-تایی‌های ممکن از پارامترها باید پوشش داده شود. در مثال ما نیاز به تولید ۷۲=۳×۲×۳×۲×۲ نمونه آزمون است. هزینه آزمون زمانی که تعداد پارامترها و مقادیر آن‌ها زیاد باشد، بسیار بالا می‌رود. علاوه بر این چون تعامل تعداد محدودی از پارامترها باعث بروز خطا می‌شود [11] تولید آزمون کامل اصلاً مقرون به صرفه نیست و باید از تکنیک‌هایی برای حل این مسئله استفاده کرد.

بنابرین به جای آزمون کامل از آزمون t-تایی استفاده می‌کنیم که مقدار t در بازه $2 \leq t \leq p$ می‌باشد. مقدار t مشخص کننده این است که دنباله آزمون ما به جای اینکه تمام ترکیبات p-تایی را پوشش دهد. باید شامل تمام ترکیبات t-تایی از پارامترها باشد. اگر مقدار t در تولید آزمون زیاد باشد. باعث به وجود آمدن تعداد زیادی نمونه آزمون بی‌ارزش می‌شود و اگر تعداد t کم باشد آن تعاملاتی که باعث وقوع خطا می‌شوند را نمی-توانیم شناسایی کنیم.

آزمون ترکیباتی تمام t-تایی‌های ممکن از پارامترها را پوشش می‌دهد. به دنباله آزمونی که در این آزمون تولید می‌شود آرایه پوشش t-تایی (Coverage Array t-way) می‌گویند. t قوه پوشش است و مقدار آن عمق پوشش را مشخص می‌کند. t یک تنظیم کلیدی برای آزمون t-تایی است که باید توسط آزمون کننده مشخص شود.

مجلـه‌ی علمی پژوهش در علوم رایانه ، شماره ۱۴، تابستان ۱۳۹۸، ص ۴۹- ۳۳

ISSN: 2745-3673
http://www.Rcsj.ir

همانطور که اشاره شد تمام حالات مثال آژانس مسافرتی ۷۲ نمونه آزمون است با اعمال 2-way تعداد کل نمونه آزمون‌ها ۹ تا می‌شود که در در جدول ۲ نشان داده شده است. این جدول تمام ترکیبات ممکن از هر دو پارامتر موجود است. برای مثال دو پارامتر بانک و هتل دارای ۲×۳=۶ حالت متفاوت دوتایی هستند که عبارتند از: (-, -, پارسیان, پاسارگاد)، (-, -, گلستان، پاسارگاد)، (-, -, پارسیان، سپه)، (-, -, گلستان, سپه)، (-, -, پارسیان, ملت)، (-, -, گلستان, ملت)، تمام این حالت‌ها را در جدول ۲ می‌توان مشاهده کرد.

جدول ۲: دنباله آزمون برای CA (9; 2, 2^3, 3^2)

ردیف	مشتری	آژانس مسافرتی	خط هوایی	هتل	بانک
۱	وحید (۰)	رخش (۰)	زاگرس (۰)	گلستان (۱)	ملت (۲)
۲	سجاد (۱)	سفیران (۱)	آسمان (۲)	پارسیان (۰)	ملت (۲)
۳	وحید (۰)	سفیران (۱)	کاسپین (۱)	گلستان (۱)	سپه (۰)
۴	سجاد (۱)	رخش (۰)	کاسپین (۱)	پارسیان (۰)	پاسارگاد (۱)
۵	وحید (۰)	رخش (۰)	آسمان (۲)	پارسیان (۰)	سپه (۰)
۶	سجاد (۱)	سفیران (۱)	زاگرس (۰)	گلستان (۱)	پاسارگاد (۱)
۷	وحید (۰)	رخش (۰)	آسمان (۲)	گلستان (۱)	پاسارگاد (۱)
۸	سجاد (۱)	سفیران (۱)	زاگرس (۰)	پارسیان (۰)	سپه (۰)
۹	سجاد (۱)	رخش (۰)	کاسپین (۱)	پارسیان (۰)	ملت (۲)

در اغلب سیستم‌های نرم‌افزاری اهمیت تعامل بین پارامترها ثابت نیست برخی از تعاملات بیشتر مستعد خطا هستند. درحالی که برخی از تعاملات دیگر تاثیر کمتری بر خطای سیستم دارند. برای اینکه بتوانیم این تعاملات متفاوت را به‌طور مؤثر شناسایی کنیم. از ساختار آرایه پوشش با قوه متغیر استفاده می‌کنیم. در این ساختار قوه‌های پوشش متفاوتی بین گروه‌های مختلف از پارامترها وجود دارد. این ساختار یک راهکار عملی برای کشف خطا در نرم‌افزارهای کاربردی است.آرایه پوشش با قوه متغیر با نماد VSCA (N, t, p, v {C}) یا VSCA (N, t, v^p, v {C}) نشان داده می‌شود. که یک آرایه پوشش {C} یک یا چند CA (t', p', v) است و p' باید زیر مجموعه‌ای از p باشد. برای فهم بیشتر VSCA مثال آژانس مسافرتی را با VSCA بسط می‌دهیم در مثال فرض می‌کنیم که سه پارامتر مشتری، آژانس مسافرتی و خط هوایی نیاز به تعامل 3-way دارند (سایر ترکیبات در حالت 2-way هستند). این حالت به شکل VSCA (N, 2, 2^3, 3^2, CA (3, 2^2, 3^1)) نشان داده می‌شود. نمونه آزمون‌های مورد نیاز برای پوشش کامل این پیکربندی در جدول ۳ نشان داده شده است.

جدول ۳: دنباله آزمون برای VSCA (12, 2, 2^3, 3^2, CA (3, 2^2, 3^1))

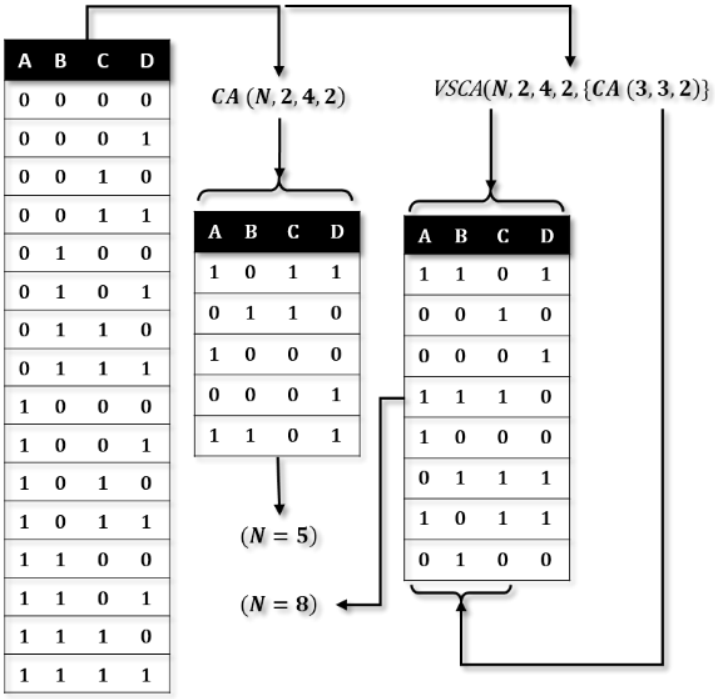
ردیف	مشتری	آژانس مسافرتی	خط هوایی	هتل	بانک
۱	وحید (۰)	رخش (۰)	زاگرس (۰)	گلستان (۱)	ملت (۲)
۲	وحید (۰)	رخش (۰)	کاسپین (۱)	گلستان (۱)	سپه (۰)
۳	سجاد (۱)	سفیران (۱)	زاگرس (۰)	پارسیان (۰)	پاسارگاد (۱)
۴	وحید (۰)	رخش (۰)	آسمان (۲)	گلستان (۱)	پاسارگاد (۱)
۵	وحید (۰)	سفیران (۱)	زاگرس (۰)	گلستان (۱)	سپه (۰)
۶	وحید (۰)	سفیران (۱)	کاسپین (۱)	پارسیان (۰)	ملت (۲)
۷	وحید (۰)	سفیران (۱)	آسمان (۲)	پارسیان (۰)	سپه (۰)
۸	سجاد (۱)	سفیران (۱)	کاسپین (۱)	گلستان (۱)	سپه (۰)
۹	سجاد (۱)	رخش (۰)	آسمان (۲)	گلستان (۱)	ملت (۲)
۱۰	سجاد (۱)	رخش (۰)	کاسپین (۱)	پارسیان (۰)	پاسارگاد (۱)
۱۱	سجاد (۱)	رخش (۰)	زاگرس (۰)	پارسیان (۰)	سپه (۰)
۱۲	سجاد (۱)	سفیران (۱)	آسمان (۲)	پارسیان (۰)	پاسارگاد (۱)

ISSN: 2745-3673
http://www.Rcsj.ir

به‌عنوان مثال دیگر، سیستمی با ۴ پارامتر ۲ مقداری را در نظر بگیرد. در حالت معمول تعداد نمونه آزمون برای این پیکربندی $2^4 = 16$ (v^p) است که در شکل۱ نشان داده شده است. اما با اعمال پیکربندی CA (N; 2, 4, 2)، تعداد نمونه آزمون‌ها به ۵ کاهش می‌یابد. معیار پوشش به‌گونه‌ای است که باید برای هر دو ستون (AB, AC, AD, BC, BD, CD) چهار مقدار (۰۰، ۰۱، ۱۰ و ۱۱) وجود داشته باشد و زمانی که تمام حالات ممکن پوشش داده شود، دنباله آزمون تکمیل می‌گردد. در حالت کلی تعداد کل پوشش‌ها برای پیکربندی هایی که پارامترهایی با مقادیر برابر دارند از رابطه (۱) بدست می‌آید.

$$Max\_Coverage = \binom{p}{t} * v^t \qquad (۱)$$

برای تشریح بیشتر آرایه پوشش با قوه متغییر، مثال VACA (N; 2, 4, 2 {CA (3, 3, 2)}) را در نظر بگیرید. در این مثال نمونه آزمون باید پیکربندی CA (N1, 2, 4, 2) بر روی چهار پارامتر A، B، C و D و پیکربندی CA (N2; 3, 3, 2) بر روی سه پارامتر A, B, C را در نظر بگیرد که نحوه تقسیم‌بندی آن در شکل۲ نشان داده شده است.



شکل ۲: نمایش پوشش آرایه با قوه ثابت و متغیر

روش‌ها

پژوهش‌های زیادی برای حل مسئله پوشش آرایه وجود دارد که به دو گروه اصلی تقسیم می‌شوند [1].

روش‌های ریاضی

روشهای محاسباتی

روش‌های ریاضی (جبری یا ترکیباتی) از برخی از ساختارهای ترکیباتی مربوط به توابع ریاضی مانند آرایه متعامد استنتاج شده‌اند. روشهای ریاضی بهینه آرایه پوشش را فقط برای پیکربندی‌های خاصی تولید می‌کنند. استراتژی Combinatorial Test Services و (TConfig) [12] بر اساس تعمیم آرایه متعامد ساخته شده‌اند. عمده‌ترین مشکل راهکارهای مبتی بر آرایه متعامد این است که فقط توانایی ساخت آرایه پوشش برای پیکربندی‌های کوچک را دارند. این محدودیت باعث استفاده بیشتر از روش‌های محاسباتی شده است.

روشهای محاسباتی عمدتاً از استراتژی‌های محاسبات محض یا تکنیک‌های فرا مکاشفه‌ای برای تولید آرایه پوشش بهینه در فضای جستجو استفاده می‌کنند.

ISSN: 2745-3673
http://www.Rcsj.ir

راهکار اغلب روش‌های محاسباتی به این صورت است که در ابتدا تمام ترکیبات ممکن با توجه به مشخصات ورودی تولید می‌شود. پس از آن نمونه آزمون‌ها برای پوشش این ترکیبات ساخته می‌شوند. تفاوت اصلی بین استراتژی‌های مختلف روش محاسباتی ساختن نمونه آزمون است. دو رویکرد اصلی برای ساخت نمونه آزمون در روش‌های محاسباتی وجود دارد.

یک-آزمون-در-یک-زمان

یک-پارامتر-در-یک-زمان

رویکرد "یک-آزمون-در-یک-زمان"، در هر تکرار یک نمونه آزمون (سطر) به دنباله آزمون اضافه می‌گردد. به عبارتی دیگر توسعه افقی[1] صورت می‌گیرد. استراتژی‌های مبتنی بر این ساختار عبارت اند از: ATEG[13]، PICT[14]، CTE-XL[15]، TVG[16]، ITCH و GTway[7] که تمامی این استراتژی‌ها در گروه استراتژی‌های مبتنی بر محاسبات جای دارند. الگوریتم Density تکامل یافته DDA است که استراتژی را قادر می‌سازد تا از ساختار VSCA پشتیبانی کند.

اولین استراتژی که با "یک-آزمون-در-یک-زمان" دنباله آزمون تولید کرد AETG بود. این استراتژی در هر سیکل تعدادی نمونه آزمون را انتخاب می‌کند و به روش حریصانه (greedy) یکی از نمونه آزمون ها را به دنباله آزمون اضافه می‌کند. در واقع نمونه آزمونی را به دنباله آزمون اضافه می‌کند که بیشترین تعداد تعاملات را پوشش دهد. این استراتژی بعدها در قالب mAETG [17]و mAETG-sat [11] توسعه داده شد.

استراتژی PICT نیز از دیگر استراتژی‌های "یک-آزمون-در-یک-زمان" که به صورت حریصانه عمل می‌کند این استراتژی تمام تعاملات را تولید می‌کند و به صورت تصادفی نمونه آزمون‌های مورد نیاز خود را انتخاب می‌کند که به دلیل مبتنی بر تصادفی بودن این استراتژی غالبا نتایج نامطلوبی را تولید می‌کند از نقاط قوت این استراتژی این است که توان تولید دنباله آزمون را برای $t > ۶$ دارد.

TVG برای تولید دنباله آزمون از سه الگوریتم T-Reduce، plus-one و Random Sets استفاده می‌کند که جزئیات این الگوریتم‌ها در دسترس نیست اما در حالت کلی الگوریتم T-Reduce نتایج بهتری را در مقایسه با دو الگوریتم دیگر دارد این استراتژی نیز دنباله آزمون را بر اساس یک-آزمون-در-یک-زمان تولید می‌کند.

از دیگر استراتژی‌های "یک-آزمون-در-یک-زمان" استراتژی Jenny [18] است این استراتژی از سرعت مناسبی برخوردار است و نتایج نسبتا قابل قبولی را برای بسیاری از پیکربندی‌ها از لحاظ بهره‌وری (efficiency) تولید می‌کند ساختار این الگوریتم به نحوی است که ابتدا تمام تعاملات 1-تایی را تولید می‌کند سپس دنباله آزمون برای تعاملات ۲-تایی تکمیل خواهد شد و این روال تا زمانی که کل تعاملات مربو به t-تایی پوشش داده شود ادامه پیدا می‌کند.

یکی از کندترین استراتژی‌های محاسباتی استراتژی ITCH است این استراتژی با جستجوی جامع (exhaustive) دنباله آزمون تولید را می‌کند که در بعضی از پیکربندی ها نتایج خوبی را از لحاظ بهره‌وری تولید می‌کند و توان تولید دنباله آزمون را تا $t = 4$ دارد. دیگر استراتژی "یک-آزمون-در-یک-زمان" استراتژی CTE_XL است این استراتژی بر اساس Classification-Tree Method (CTM) دنباله آزمون را تولید می‌کند در این استراتژی دامنه ورودی بر اساس ویژگی ها به زیرمجموعه‌های مختلفی تقسیم و سپس نمونه آزمون با ترکیب این زیرمجموعه‌ها تولید و به دنباله آزمون اضافه می‌گردد این استراتژی نیز غالبا تا $t = 3$ توان تولید دنباله آزمون را دارد.

یکی از قدرتمندترین استراتژی های محاسباتی، استراتژی GTWay است این استراتژی که بر اساس "یک-آزمون-در-یک-زمان" دنباله آزمون تولید می‌کند ابتدا تمامی حالات نمونه آزمون را به صورت ساختار بیتی ذخیره می‌کند و برای دسترسی سریع‌تر به ترکیبات مربوط به نمونه آزمون‌ها از یک جدول ایندکس استفاده می‌کند این استراتژی یکی از معدود استراتژی‌هایی است که از هر دو لحاظ کارایی و بهره‌وری نتایج بسیار مطلوبی را تولید می‌کند استراتژی GTWay توان تولید تا $t = 12$ را دارد.

استراتژی‌های مبتنی بر هوش مصنوعی از دیگر استراتژی‌های "یک-آزمون-در-یک-زمان" هستند. این استراتژی‌ها دارای قدرت بالایی در تولید دنباله آزمون با اندازه آرایه کمینه می‌باشند و معمولا سرعت پایینی دارند. اما از لحاظ زمان، ضعیف‌تر عمل می‌کنند. مهم‌ترین این الگوریتم ها GA، SA، TS، ACA، Cuckoo Search، PSO و HS هستند. یکی از معیارهای دسته‌بندی استراتژی‌ها در تولید دنباله آزمون، قوه تعامل است.

در ابتدا Stardom در [19] الگوریتم‌های SA، GA و TS برای تعامل دودویی[2] پیاده‌سازی کرد. این استراتژی‌ها به دلیل داشتن ساختار پیچیده نتایج بسیار مطلوبی را استخراج می‌کند. نتایج SA بهتر از TS و TS بهتر از GA است. این استراتژی‌ها در VSCA قادر به تولید دنباله آزمون نیستند و فقط در CA دنباله آزمون تولید می‌کند. الگوریتم SA در [11] توسعه داده شد به نحوی که قادر است تعامل 3-way را پشتیبانی کند. دو الگوریتم GA و ACA در [11] ارتقا داده شدند تا 3-way را پشتیبانی کند. این استراتژی‌ها فقط ساختار CA را پشتیبانی می‌کنند.

ISSN: 2745-3673
http://www.Rcsj.ir

نتایج نشان می‌دهد در این تعامل استراتژی SA قوی‌تر از GA و ACA است. برای پشتیبانی از VSCA الگوریتم SA در [11] توسعه یافت. نتایج این استراتژی برای قوه ۲ و ۳ در دسترس است و در ادامه الگوریتم ACA را برای ساختار VSCA توسعه داد و آن را ACS[۳] نامید. نتایج منتشر شده در این مقاله نشان می‌دهد استراتژی SA بسیار قوی‌تر از ACS است. نتایج در این استراتژی بر روی قوه ۲ و ۳ (2-way, 3-way) متمرکز است. استراتژی PSTG اولین استراتژی مبتنی بر هوش مصنوعی (الگوریتم PSO) بود که قادر است تا قوه ۶ را پشتیبانی کند. نتایج این استراتژی برای t<=3 ضعیف‌تر از دیگر استراتژی‌های مبتنی بر هوش مصنوعی است ولی سرعت بیشتری دارد.

این استراتژی راهکار جدیدی برای محاسبه وزن نمونه آزمون ارائه داد که مستلزم نگه‌داری ساختمان داده‌های بزرگ برای محاسبه نمونه آزمون‌ها است. استراتژی PSTG ساختار VSCA را نیز پشتیبانی می‌کند که آن را VS-PSTG نامیده‌اند. استراتژی CS نیز تا قوه ۶ دنباله آزمون را فقط برای ساختار CA تولید می‌کند. ایده اصلی در این استراتژی کاهش فضای جستجوی برای نمونه آزمون با استفاده از الگوریتم فاخته است که نتایج مشابهی را با الگوریتم PSTG با سرعت بیشتر تولید می‌کند. استراتژی HSS دیگر استراتژی قدرت‌مند در VSCA است که تولید دنباله آزمون تا قوه ۱۵ را پشتیبانی می‌کند. نتایج منتشر شده در این مقاله نشان می‌دهد که قدرت این استراتژی در بسیاری از پیکربندی‌ها از VS-PSTG بیشتر است.

علاوه بر [4]، استراتژی های PWiesGen، GAPTS و PWiesGenPM نیز با استفاده از الگوریتم ژنتیک دنباله آزمون تولید می‌کنند. این استراتژی ها به دلیل ساختار پیچیده تا قوه ۲ را برای CA پشتیبانی می کنند که GAPTS قوی تر از PWiesGen و PWiesGenPM قوی تر از PWiesGen است. در بعدها تغییراتی در ساختار الگوریتم ژنتیک ایجاد شده است که می تواند با تعداد تکرار کمتری نسبت به PWiesGen به نتیجه برسد. همچنین استراتژی PWiesGen ارتقا داده شد به نحوی که ساختار VSCA را پشتیانی می کند این روش تا قوه ۶ را پشتیبانی می کند که آن را PWiesGen-VSCA تامیده است. نتایج منتشر شده در این مقاله برای ساختار VSCA به گونه ای است که که در ساختار اصلی تا قوه ۲ و در زیر ساختار تا قوه ۶ را پشتیبانی میکند و به همین دلیل است که خود را جز دسته ی pairwise قرار داده است. دیگر الگوریتم مبتنی بر GA است که آن را GS (genetic strategy) [1] نامیده اند. در این راهکار با تغییر تابع محاسبه وزن نمونه آزمون و کاهش زمان محاسبه آن توانسته است سرعت استراتژی را بالا ببرد. نحوه ذخیره‌سازی نمونه‌های آزمون تا حدودی از استراتژی GTWay استفاده می‌کند. از دیگر عوامل افزایش سرعت می‌توان به کاهش پیچیدگی‌های مربوط به الگوریتم ژنتیک، از جمله تابع CrossOver، Mutation و حذف تابع مرتب سازی اشاره کرد.

سرعت GS با این تغییرات بسیار افزایش یافته است و حتی در بسیاری از پیکربندی‌ها از استراتژی‌های مبتنی بر محاسبات محض بهتر عمل می‌کند. این راهکار با کاهش اندازه آرایه‌های محاسبه وزن و الگوریتم ژنتیک توانسته است دنباله آزمون را تا t = ۲۰ تولید کند همچنین از نظر اندازه آرایه نیز نتایج مطلوبی را بدست می‌آورد.

استراتژی HHH [20] اولین استراتژی است که از HHH (High Level Hyper-Heuristic) استفاده می‌کند در این استراتژی به جای استفاده از یک الگوریتم فرامکاشفه‌ای از چهار الگوریتم فرامکاشفه ای که عبارتند از Teaching Learning based Optimization (TLBO), Particle Swarm Optimization (PSO), Global Neighborhood Algorithm (GNA), and Cuckoo Search Algorithm (CS) استفاده می‌کند. در حقیقت این استراتژی از الگوریتم ممنوعه استفاده می‌کند و در هر مرحله بر اساس سه اپراتور improvement, diversification and intensification یکی از این چهار الگوریتم را برای تولید نمونه آزمون انتخاب می‌کند. این استراتژی نیز نتایج بسیار خوبی را از لحاظ بهره‌وری دارد.

سایر استراتژی‌های مبتنی بر الگوریتم PSO در تولید دنباله آزمون کمینه که همگی در [9] پیاده‌سازی شده‌اند عبارتند از: TVAC, CLPSO, APSO, DMS-PSO, DPSO و CPSO. یکی از نقاط ضعف استراتژی های مبتنی بر PSO در تولید دنباله آزمون این است که با اعمال سرعت لزوما مقدار جدید بهبود داده نخواهد شد برای غلبه بر این مشکل استراتژی DPSO با ارائه راهکار مناسب توانسته است نتایج بسیار مطلوبی را از لحاظ بهره‌وری بدست آورد نتایج مربوط به این استراتژی در مقاله مربوطه تا t = ۴ است اما می‌تواند t > ۶ را نیز پشتیبانی کند.

یکی دیگر از راهکارهای موثر در تولید دنباله آزمون [21] FSAPSO[۴] است. همانطور که از نامش پیداست این استراتژی مبتنی بر الگوریتم PSO است در این استراتژی مقادیر پارامترهای C1، C2 و W به صورت فازی تعیین می‌گردد این استراتژی تا t = ۴ دنباله آزمون تولید می‌کند و از لحاظ بهره وری نتایجی بهتر از DPSO و CS را در بسیاری از ساختارها دارد. طبیعی است که استراتژی FSAPSO بدلیل استفاده از ابزار فازی در متلب کارایی مناسبی ندارد.

ISSN: 2745-3673
http://www.Rcsj.ir

در [22] نیز با ترکیب الگوریتم جستجوی تپه نوردی و الگوریتم جستجوی خفاش (HC-BAT)، دنباله آزمون را برای پیکربندی‌های مختلف تولید می‌کند در این راهکار هیچگونه داده ای برای برای محاسبه وزن ذخیره نمی‌شود و محاسبه وزن بر اساس دنباله آزمون تولید شده محاسبه می‌شود این استراژی توان تولید دنباله آزمون را تا قوه ۱۰ دارد.

و در [23] با استفاده از الگوریتم بهینه‌سازی مبتنی بر آموزش و یادگیری (TLBO) که یکی از الگوریتم‌های بهینه‌سازی کارآمد و جدید است از ساختار ساده آن، توانایی جستجوی بالا و تنوع‌پذیری در تولید جواب بهره برده است و توانسته است دنباله آزمون تا قوه ۱۵ راتولید کند نتایج این استراتژی نیز نشان می‌دهد که از لحاظ کارایی و بهره‌وری نیز بسیار مناسب است.

در رویکرد "یک-پارامتر-در-یک-زمان" ابتدا با دو پارامتر الگوریتم شروع به کار می‌کند. دنباله آزمون به صورت سطری (توسعه افقی) اضافه می‌گردد. اگر پوشش کامل شود الگوریتم با بیش از دو پارامتر شروع به کار می‌کند (توسعه عمودی) و تا آخر ادامه می‌یابد. استراتژی IPO در این دسته قرار دارد. استراتژی‌های دیگری بر اساس این استراتژی تکامل یافتند که عبارتند از: MIPOG، IPOG-F IPO-S، IPOG , IPOG-D و ParaOrder. تنها دو استراتژی ParaOrder و IPOG می‌توانند ساختار VSCA را پشتیبانی کنند. استراتژی ParaOrder تکامل یافته IPO است [1].

مقایسه روش‌ها

ارزیابی در استراتژی‌های تولید دنباله آزمون به طور کلی به سه دسته تقسیم می‌شوند: (۱) ارزیابی از لحاظ اندازه آرایه دنباله آزمون (یا بهره-وری[۵])، (۲) زمان تولید دنباله آزمون (کارایی[۶]) و قوه تعامل [2]. ارزیابی با معیار اندازه آرایه فقط به مراحل اجرای الگوریتم بستگی دارد و مستقل از سخت افزار و سیستم عامل می‌باشد. ارزیابی زمانی وابسته به سخت افزار و سیستم عامل است. لذا سعی کرده‌ایم استراتژی‌های که در دسترس هستند را بر روی سیستم عامل خود اجرا نموده و از لحاظ کارایی با یکدیگر مقایسه نماییم مشخصات بستر سخت‌افزاری برای اجرای روش‌های موجود عبارت است از: Windows 7، CPU 2.20GHz core™ i7QM و RAM 6GB. در جداول مربوط به ارزیابی، NA[۷] به مفهوم عدم انتشار در مقالات و NS[۸] پشتیبانی نکردن استراتژی از پیکربندی مربوطه است [3]. بهترین نتایج در هر سطر، ضخیم[۹] نمایش داده شده است.

بهره‌وری

مهمترین معیار برتری تولید دنباله آزمون بهره‌وری راهکار است به عبارت دیگر هر الگوریتمی که بتواند دنباله آزمون با تعداد سطر کمتر برای یک پیکربندی تولید کند آن الگوریتم در پیکربندی مربوطه قوی‌تر است به شرطی که دنباله آزمون تولید شده پوشش کامل را ارائه دهد و حداکثر زمان برای تولید آن ۲۴ ساعت باشد با توجه به تعداد زیاد پیکربندی‌های مورد ارزیابی در مقالات منتشر شده، برای نشان دادن توان الگوریتم‌های موجود سعی کردیم پیکربندی‌های که بیشتر در مقالات استفاده شده است را در قالب دو جدول نشان دهیم. در جدول ۴ به بررسی پیکربندی $CA(N; t, 7, v)$ که $۵ \le v \le ۲$ و $۴ \le t \le ۲$ است. همانطور که ملاحظه می‌شود استراتژی‌های مبتنی برهوش مصنوعی بسیار قوی-تر از استراتژی‌های مبتنی بر محاسبات می‌باشند در بین استراتژی‌های مبتنی بر محاسبات ITCH بسیار قوی عمل می‌کند و در چند پیکربندی بهترین نتایج را تولید می‌کند. در میان استراتژی‌های مبتنی بر هوش مصنوعی بهترین نتایج را DPSO تولید می‌کند همانطور که مشاهده می‌شود این استراتژی نتایج بسیار مناسبی را دارد و می‌توان گفت این استراتژی قوی‌ترین استراتژی مبتنی بر هوش مصنوعی برای $t \ge ۳$ در بین استراتژی‌های موجود در جدول ۴ را تولید می‌کند.

جدول ۵ پیکربندی‌های مستقل که بیشترین استفاده در مقالات دارند را جمع آوری کرده‌ایم در این جدول تقریبا تمامی استراتژی‌ها موجود را مورد بررسی قرار می‌دهیم در این پیکربندی‌ها نیز استراتژی‌های مبتنی بر هوش مصنوعی بسیار قوی‌تر عمل می‌کنند باتوجه به اینکه نتایج بعضی از استراتژی‌ها در پیکربندی‌ها مختلف در دسترس نیست نمی‌توان بصورت قطعی در خصوص برتری یک الگوریتم بر سایر الگوریتم‌ها اظهار نظر کرد اما به‌طور کلی می‌توان گفت استراتژی PWiesGenPM در $t = 2$ و SA در $t = 3$ بهترین نتایج را تولید می‌کنند همانطور که در بخش ۴-۳ بررسی می‌شود نقطه ضعف این دو الگوریتم این است که PWiesGenPM تا $t = 2$ و SA تا $t = 3$ دنباله آزمون تولید می‌کند و توان تولید برای قوه‌های بالاتر را ندارد. در $t = 4$ استراتژی HHH بهترین نتایج را تولید می‌کند اما با توجه به نتایج قبل مشخص است که در صورت در دسترس بودن نتایج مربوط به DPSO این استراتژی بسیار قوی‌تر است. پس می‌توان به عنوان خروجی این بخش به این نتیجه رسید که در $t = 2$ استراتژی PWiesGenPM، در $t = 3$ استراتژی SA و در $t \ge 4$ استراتژی DPSO بهترین نتایج را از لحاظ بهره‌وری تولید می-کنند لازم بذکر است که استراتژی ITCH برای $t \le 4$ نتایج بسیار مطلوبی را دارد و بعضا از استراتژی‌های مبتنی بر هوش مصنوعی هم بهتر عمل خواهد کرد.

ISSN: 2745-3673
http://www.Rcsj.ir

جدول۴: اندازه آرایه برای CA (N; t, 7, v) با ۵ ≥ v ≥ ۲ و ۴ ≥ t ≥ ۲

t	v		استراتژی‌های مبتنی بر هوش مصنوعی								استراتژی‌های مبتنی بر محاسبات						
		HHH	HSS	PSTG	CS	DPSO	FSAPSO	TLBOS	GS	HC-BAT	Jenny	TConfig	ITCH	PICT	TVG	CTE-XL	IPOG
۲	۲	۷	۷	۶	۶	۷	۶	۶	۶	۶	۸	۷	۶	۷	۷	۸	۸
	۳	۱۴	۱۴	۱۵	۱۵	۱۴	۱۵	۱۵	۱۴	۱۵	۱۶	۱۵	۱۵	۱۶	۱۵	۱۶	۱۷
	۴	۲۳	۲۵	۲۶	۲۵	۲۴	۲۵	۲۴	۲۴	۲۵	۲۸	۲۸	۲۸	۲۷	۲۷	۳۰	۲۸
	۵	۳۵	۳۵	۳۶	۳۷	۳۴	۳۵	۳۷	۳۶	۲۶	۳۷	۴۰	۴۵	۴۰	۴۲	۴۲	۴۲
۳	۲	۱۵	۱۲	۱۳	۱۲	۱۵	۱۵	۱۲	۱۲	۱۳	۱۴	۱۶	۱۳	۱۵	۱۵	۱۵	۱۹
	۳	۴۹	۵۰	۵۰	۴۹	۴۹	۴۸	۴۹	۴۹	۴۹	۵۱	۵۵	۴۵	۵۱	۵۵	۵۴	۵۷
	۴	۱۱۲	۱۲۱	۱۱۶	۱۱۷	۱۱۲	۱۱۸	۱۱۶	۱۱۶	۱۱۶	۱۲۴	۱۱۲	۱۱۲	۱۲۴	۱۳۴	۱۳۵	۲۰۸
	۵	۲۱۶	۲۲۳	۲۲۵	۲۲۳	۲۱۶	۲۳۹	۲۲۴	۲۲۱	۲۲۵	۲۳۶	۲۳۹	۲۲۵	۲۴۱	۲۶۰	۲۶۵	۲۷۵
۴	۲	۳۱	۲۹	۲۹	۲۷	۳۴	۳۰	۲۷	۲۷	۲۷	۳۱	۳۶	۴۰	۳۲	۳۱	NA	۴۸
	۳	۱۴۸	۱۵۵	۱۵۵	۱۵۵	۱۵۰	۱۵۳	۱۵۴	۱۵۳	۱۵۴	۱۶۹	۱۶۶	۲۱۶	۱۶۸	۱۶۷	NA	۱۸۵
	۴	۴۸۲	۵۰۰	۴۸۷	۴۸۷	۴۷۲	۴۷۲	۴۸۶	۴۸۶	۴۹۰	۵۱۷	۵۶۸	۷۰۴	۵۲۹	۵۵۹	NA	۵۰۹
	۵	۱۱۵۳	۱۱۷۴	۱۱۷۶	۱۱۷۱	۱۱۴۸	۱۱۶۲	۱۱۶۹	۱۱۷۳	۱۱۸۰	۱۲۴۸	۱۳۲۰	۱۷۵۰	۱۲۷۹	۱۳۸۵	NA	۱۳۴۹

ISSN: 2745-3673
http://www.Rcsj.ir

کارایی

همانطوری که قبلا اشاره شد کارایی یا زمان تولید دنباله آزمون به سخت افزار و سیستم عامل بستگی دارد پس تنها راه مقایسه این استراتژی‌ها اجرا بر روی یک سیستم با سیستم عامل واحد است. برای رسیدن به این هدف سعی کردیم استراتژی‌های در دسترس را بر روی سیستم خود اجرا نماییم که مشخصات بستر سخت‌افزاری عبارتند از: Windows 7 ، core™ i7QM CPU 2.20GHz و 6GB RAM. جدول۶ استراتژی‌های در دسترس را با ۲ مقدار زمان و اندازه آرایه نشان می‌دهد همانطوری که ملاحظه می‌شود استراتژی IPOG و IPOG-D بیسیار سریع‌تر از سایر استراتژی‌ها عمل می‌کنند و کندترین استراتژی نیز DPSO است استراتژی‌های Jenny و PICT نیز کارایی مناسبی در تولید دنباله آزمون دارند اما TConfig سرعت مناسبی در تولید دنباله آزمون ندارد.

قوه تعامل

سومین معیار سنجش برتری استراتژی‌های تولید دنباله آزمون قوه تعامل است افزایش این عامل بیشتر به نحوه ذخیره سازی نمونه آزمون و همچنین سرعت تولید دنباله آزمون بستگی دارد. ساختمان داده‌های متفاوتی برای محاسبه وزن نمونه‌آزمون وجود که حجم این ساختمان داده ها متفاوت هست و هرچه پیچیدگی بیشتر باشد قوه تعامل کاهش میآبد. برای ذخیره سازی حالات پوشش داده نشده (حالت اولیه) بهترین راهکار استفاده از ساختار بیتی است منظور از ساختار بیتی این است که برای هر نمونه آزمون پوشش داده نشده یک بیت در نظر پرفته می‌شود در این روش برخلاف روش هایی مانند CS و PSTG برای v^t نمونه آزمون یک سطر در نظر گرفته می‌شود و به تعداد ترکیبات $\binom{p}{t}$ سطر ایجاد می‌گردد این کار ما را از سویچ کردن بین ماتریس‌ها بی نیاز می‌کند.

ISSN: 2745-3673
http://www.Rcsj.ir

جدول ۵: اندازه آرایه برای پیکربندی‌های مختلف

	مبتنی بر محاسبات										مبتنی بر هوش مصنوعی															
	TVG	TConfig	PICT	Jenny	AllPairs	IPOG	TCG	AETG	mAETG	GS	TLBOS	DPSO	FSAPSO	ACO	GA	SA	CS	PSTG	HSS	HHH	PWiseGenPM	PWiseGen	GAPST	HC-BAT	TS
CA (N; 2, 3^3)	NA	۱۰	۱۰	۹	۹	۹	NA	NA	NA	۹	۹	۹	۹	NA	NA	NA	NA	NA	NA	NA	۹	۹	NA	۱۰	NA
CA (N; 2, 3^4)	۱۱	۹	۱۲	۱۱	۹	۹	NA	۹	۹	۹	۹	۹	NA	۹	۹	۹	۹	۹	۹	۹	۹	۹	۹	۹	NA
CA (N; 2, 3^{13})	۱۹	۱۵	۲۰	۱۸	۱۷	۱۹	۲۰	۱۵	۱۷	۱۸	۱۸	۱۷	۱۶	۱۷	۱۷	۱۶	۲۰	۱۷	۱۸	۱۷	۱۵	۱۶	۱۵	۱۸	NA
CA (N; 2, 5^{10})	۵۱	۴۸	۴۷	۴۵	۴۷	۴۵	NA	NA	NA	۴۱	۲۱۲	NA	NA	NA	NA	NA	NA	۴۵	۴۳	۴۲	۴۰	۴۳	NA	۴۲	NA
CA (N; 2, 10^{20})	NA	۲۳۱	۲۱۶	۱۹۳	۱۹۷	۲۲۷	۲۱۸	۱۸۰	NA	۲۱۱	۲۰۵	۱۴۶	NA	NA	NA	NA	NA	NA	NA	NA	۱۹۴	۲۲۴	NA	۲۱۰	NA
CA (N; 2, 2^{100})	NA	۱۴	۱۶	۱۶	۱۴	۱۶	۱۶	۱۰	NA	۱۵	۱۵	NA	NA	۱۴	۱۴	NA	NA	NA	NA	NA	۱۰	۱۱	۱۰	۱۵	NA
CA (N; 2, 4^5 3^4)	NA	۲۸	NA	۲۶	۲۲	۲۴	NA	NA	NA	۲۱	۲۲	NA	NA	NA	NA	۱۹	NA	NA	NA	NA	۱۹	۲۱	NA	۲۱	۱۹
CA (N; 2, $5^1 3^8 2^2$)	۲۲	۲۱	۲۰	۲۳	۲۰	۱۹	۲۰	۱۹	۲۰	۲۰	۲۱	NA	۱۸	۱۶	۱۵	۱۵	۲۱	۲۱	۲۱	۲۰	۱۵	۱۶	NA	۲۱	۱۵
CA (N; 2, $8^2 7^2 6^2 5^2$)	NA	NA	۵۶	۵۷	۵۴	۵۳	NA	NA	NA	۵۵	۵۵	NA	NA	NA	NA	۴۹	NA	NA	NA	NA	۴۹	۵۰	NA	۵۶	NA
CA (N; 2, $7^2 6^2 4^2 3^2 2^2$)	NA	۶۴	۸۰	۷۶	۶۴	۷۲	NA	NA	NA	۷۱	۷۰	NA	NA	NA	NA	۶۴	NA	NA	NA	NA	۶۴	۷۲	NA	۶۷	۶۴

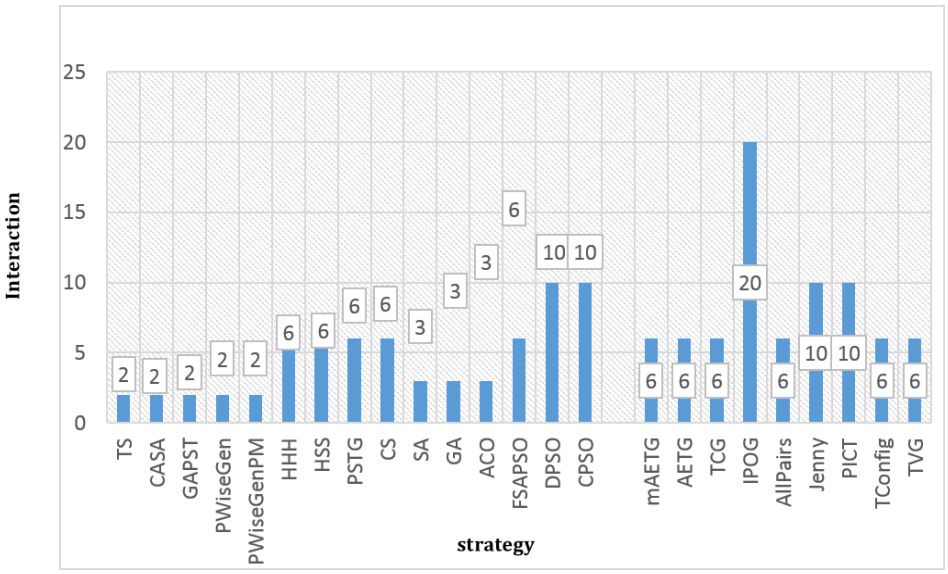
ISSN: 2745-3673
http://www.Rcsj.ir

CA (N; $2,7^1 6^1 5^1 4^6 3^8 2^3$)	NA	NA	NA	NA	NA	۴۸	۵۰	۴۸	۵۱	۴۲	۴۲	۴۲	۴۲	NA	۴۶	۴۵	۴۴	۴۵	NA	۴۳	NA	۵۰	NA	NA	۵۱
CA (N; 2, $6^1 5^1 4^6 3^8 2^3$)	NA	NA	NA	NA	NA	۳۶	۳۸	۳۹	۴۳	۳۰	۳۳	۳۲	۳۵	NA	۴۱	۴۱	۳۵	۳۴	NA	۳۵	NA	NA	NA	NA	۴۳
CA (N; 3, 3^6)	NA	۴۳	NA	NA	NA	۳۳	۳۹	۴۲	۴۳	۳۳	۳۳	۳۳	۳۶	۳۳	۴۲	۴۱	۳۸	۴۷	NA	۵۳	NA	۵۱	NA	NA	۴۹
CA (N; 3, 4^6)	NA	۱۰۵	NA	NA	NA	۶۴	۷۰	۱۰۲	۱۰۵	۶۴	۶۴	۶۴	۷۵	NA	۱۰۲	۱۰۰	۷۷	۱۰۵	NA	۶۴	NA	۱۱۲	NA	NA	۱۲۳
CA (N; 3, 6^6)	NA	۳۳۹	NA	NA	NA	۳۲۵	۳۳۶	۳۳۸	۳۵۰	۳۰۰	۳۳۱	۳۳۰	۳۳۲	۳۲۱	۳۳۵	۳۳۲	۳۳۰	۳۴۳	NA	۳۸۲	NA	۳۷۳	NA	NA	۴۰۷
CA (N; 3, 5^7)	NA	۲۲۵	NA	NA	NA	۲۱۷	۲۳۶	۲۲۹	۲۵۳	۲۰۱	۲۱۸	۲۱۸	۲۱۹	۲۱۵	۲۲۳	۲۲۸	۲۱۸	۲۲۹	NA	۲۷۴	NA	۲۳۶	NA	NA	۲۷۱
CA (N; 4, $3^4 4^5$)	NA	NA	NA	NA	NA	۴۲۷	۴۳۶	۴۴۷	NA	NS	NS	NS	NA	NA	۴۴۰	۴۳۹	NA	NA	NA	۴۶۳	NA	۴۵۷	NA	NA	۴۸۷
CA (N; 4, $5^1 3^8 2^2$)	NA	NA	NA	NA	NA	۲۸۳	۲۸۶	۲۹۲	NA	NS	NS	NS	NA	NA	۲۹۲	۲۹۰	NA	NA	NA	۳۲۴	NA	۳۰۳	NA	NA	۳۱۳

ISSN: 2745-3673
http://www.Rcsj.ir

جدول۶: اندازه آرایه و زمان برای CA (N; 3, p, 3) با ۱۲ ≥ p ≥ ۴

p	Jenny Time/N	TConfig Time/N	PICT Time/N	IPOG-D Time/N	IPOG Time/N	GS Time/N	HC-BAT Time/N	TLBOS Time/N	DPSO Time/N
۴	۰,۰۱/۳۴	۰,۰۷/۳۲	۰,۰۴/۳۴	۰,۰۰۱/۲۷	۰,۰۰۱/۳۲	۰,۴/۲۷	۰,۹/۲۸	۰,۴/۲۷	۲,۰۰/۲۷
۵	۰,۰۳/۴۰	۰,۱۰/۴۰	۰,۰۹/۴۳	۰,۰۰۱/۴۵	۰,۰۰۱/۴۱	۰,۶۳/۳۸	۱,۱/۳۸	۰,۹۶/۳۸	۷,۰۰/۴۱
۶	۰,۰۶/۵۱	۰,۳۱/۴۸	۰,۱۳/۴۸	۰,۰۰۱/۴۵	۰,۰۰۱/۴۶	۰,۷۳/۴۳	۳,۱۰/۴۳	۱,۰۵/۴۳	۱۰,۰۰/۳۳
۷	۰,۰۷/۵۱	۰,۴۳/۵۵	۰,۲۳/۵۱	۰,۰۰۱/۵۰	۰,۰۰۱/۵۵	۰,۷۸/۴۹	۶,۳۵/۴۹	۲,۶۳/۴۹	۲۳,۰۰/۴۸
۸	۰,۰۷/۵۸	۱,۲۳/۵۸	۰,۳۶/۵۹	۰,۰۰۱/۵۰	۰,۰۱۶/۵۶	۰,۸۶/۵۴	۹,۲۳/۵۳	۳,۲۱/۵۴	۳۶,۰۰/۵۲
۹	۰,۰۸/۶۲	۱,۷۲/۶۴	۰,۵۷/۶۳	۰,۰۰۱/۷۱	۰,۰۱۶/۶۳	۱,۱۱/۵۸	۱۵,۹۶/۵۷	۴,۹۲/۵۸	۵۵,۰۰/۵۶
۱۰	۰,۱۰/۶۵	۲,۸۴/۶۸	۰,۶۴/۶۵	۰,۰۰۱/۷۱	۰,۰۱۶/۶۶	۱,۲۴/۶۱	۲۶,۷۴/۶۲	۶,۲۳/۶۱	۸۱,۰۰/۵۹
۱۱	۰,۱۲/۶۵	۳,۹۳/۷۲	۰,۷۰/۷۰	۰,۰۰۱/۷۶	۰,۰۱۶/۷۰	۱۰,۲۸/۶۳	۳۹,۵۴/۶۴	۹,۵۶/۶۳	۱۱۵,۰۰/۶۳
۱۲	۰,۱۸/۶۸	۵,۲۴/۷۷	۰,۷۹/۷۲	۰,۰۰۱/۷۶	۰,۰۱۶/۷۳	۱۰,۵۳/۶۷	۵۸,۳۲/۶۶	۱۳,۲۰/۶۶	۱۵۷,۰۰/۶۵



شکل۳: ارزیابی پوشش قوه تعامل استراتژی‌های فرامکاشفه‌ای و محاسبات محض

نتیجه‌گیری و پیشنهاد

استراتژی‌های مبتنی بر هوش مصنوعی به دلیل داشتن ساختار پیچیده و همچنین تکرارهای بسیار زیاد، نتایج بسیار خوبی را در تولید دنباله آزمون با کمترین نمونه آزمون ممکن را دارند. این ساختار پیچیده رابطه معکوسی با t دارد یعنی هرچه استراتژی پیچیدگی بالایی داشته باشد قوه کمتری را پوشش می دهد به عنوان مثال میتوان از الگوریتم های GA، SA و ACO نام برد که غالبا تا t = ۳ توان تولید دنباله آزمون را دارند استراتژی های قدرتمندی چون PSTG و CS توانستند با کاهش پیچدگی، تغییر ساختمان داده و افزایش سرعت تا t = ۶ را پشتیبانی کنند.استراتژی‌های CPSO و DPSO تا t = ۱۰ و استراتژی HSS و TLBOS تا t = ۱۰ و در نهایت استراتژی GS تا t = ۱۰ را پشتیبانی می‌کند.

استراتژی‌های محاسباتی مانند PICT، Jenny و IPOG نیز از آن دسته استراتژی هایی هستند که که قادر به تولید دنباله آزمون برای تولید t > ۶ هستند این استراتژی‌های قابلا از بهره‌وری بالایی برخوردار نیستند اما کارایی مناسبی را دارند.

استراتژی GS از معدود استراتژی هایی است که می‌تواند دنباله آزمون را تا t = ۲۰ تولید کند و خروجی در بخش ارزیابی نشان دهنده آن است که بهره‌وری آن قابل رقابت با بسیاری از استراتژی های مبتنی بر هوش مصنوعی است. استراتژی SA از نظر بهره‌وری یکی از قوی‌ترین استراتژی‌های تولید دنباله آزمون برای t < ۴ است و برای قوه‌های کمتر از ۱۰ نیز استراتژی DPSO بسیار توانمند است

ISSN: 2745-3673
http://www.Rcsj.ir

اما توان تولید قوه‌های بالا از ۱۰ را ندارد. از لحاظ کارایی نیز استراتژی‌های مبتنی بر محاسبات خانواده IPOS نیز بسیار توانمند هستند اما همانطور که از نتایج نمایان است این استراتژی‌ها از لحاظ بهره‌وری بسیار ضعیف عمل می‌کنند. لذا خلاء وجود یک استراتژی که توان تولید قوه بالا با بهره‌وری و کارایی مناسب را داشته باشد، به چشم می‌خورد.

استراتژی TGway و GS از ساختار بیتی استفاده کرده که قادر خواهد بود تا $t = ۱۲$ را پشتیبانی کند و استراتژی‌های CS و PSTG تا قوه ۶ توان دنباله آزمون را دارند. دیگر استراتژی قدرتمند در قوه تعامل استراتژی HSS است نتایج این استراتژی تا قوه ۱۵ در مقاله مربوطه نشان داده شده است یک معیار برای پذیرفته شدن دنباله آزمون زمان تولید کمتر از ۲۴ ساعت می‌باشد که در مقاله HSS این محدودیت حذف شده است این امر می‌تواند بدان معنا باشد که نتایج بدست آمده در این مقاله به بیش از ۲۴ ساعت زمان نیاز دارد از طرفی پیکربندی‌های ارزیابی شده در مقاله برای tهای بزرگ به نحوی انتخاب شده‌اند که با محاسبه نیز بدست می‌آیند به عنوان مثال برای پیکربندی $CA(15, 3^{15})$ زیر مجموعه $\{VSCA (N, 2, 3^{20} 10^2, \{ CA(15, 3^{15})\}$ زمان‌بر خواهد بود به عبارت دیگر جز اصلی پیکربندی است. اگر الگوریتم به نحوی طراحی شده باشد که دنباله آزمون سطر تکراری (سطر با وزن صفر) را نپذیرد و تعداد پارامترها با قوه تعامل یکسان باشد (مانند پیکربندی $CA(15, 3^{15})$ که در آن تعداد پامترها و قوه تعامل برابر ۱۵ است) تعداد سطر دنباله آزمون برابر با v^t خواهد بود برای مثال فوق اندازه آرایه برابر ۱۴۳۴۸۹۰۷ نمونه آزمون است به نظر می‌رسد با توجه به ساختار HSS تولید این مقدار ماه‌ها زمان نیاز داشته باشد. با این وجود سعی شده است قدرت قوه تعامل هریک از استراتژی‌های موجود را در شکل ۳ نشان دهیم.

ISSN: 2745-3673
http://www.Rcsj.ir

منابع و مراجع

[1] R.N. Kacker, D. Richard Kuhn, Y. Lei and J.F. Lawrence, Combinatorial testing for software: An adaptation of design of experiments, Measurement, vol. 46, pp. 3745–3752, 2013.

[2] B. S. Ahmed, T. S. Abdulsamad and M. Y. Potrus, "Achievement of minimized combinatorial test suite for configuration-aware software functional testing using the Cuckoo Search algorithm", Information and Software Technology, vol. 66, pp. 13-29, 2015.

[3] S. Singh, A. Kaur, K. Sharma and S. Srivastava, "Software Testing Strategies and Current Issues in Embedded Software Systems", International Journal of Scientific & Engineering Research, vol. 4, 2013.

[4] S. Nidhra, J. Dondeti, "Black box and white box testing techniques–a literature review", International Journal of Embedded Systems and Applications (IJESA), vol.2, no.2, 2012.

[5] W. Afzal, R. Torkar and R. Feldt, "A systematic review of search-based testing for non-functional system properties", Information and Software Technology, vol. 51, pp. 957-967, 2009.

[6] P. Mitra, S. Chatterjee and N. Ali, "Graphical Analysis of MC/DC using Automated Software Testing", Electronics Computer Technology (ICECT), 3rd International Conference on vol. 3, 2011.

[7] T. Murnane, K. Reed, "On the Effectiveness of Mutation Analysis as a Black Box Testing Technique", Software Engineering Conference, pp. 12-20, 2001.

[8] P. Tripathy, K. Naik, "Software Testing and Quality Assurance: Theory and Practice", John Wiley & Sons, 2008.

[9] A. Rahman A. Alsewari, K. Z. Zamli, "Design and implementation of a harmony-search-based variable-strength t-way testing strategy with constraints support", Information and Software Technology, vol. 54, pp. 553–568, 2012.

[10] C.J. Colbourn, G. Kéri, P.P. Rivas Soriano, J.C. Schlage-Puchta, "Covering and radius-covering arrays: Constructions and classification", Discrete Applied Mathematics, vol. 158, pp. 1158-1180, 2010.

[11] B. S. Ahmeda, K. Z. Zamlia and C. P. Lim, "Application of Particle Swarm Optimization to uniform and variable strength covering array construction", Applied Soft Computing, vol. 12, pp. 1330–1347, 2012.

[12] C. Yilmaz, M.B. Cohen and A. Porter, "Covering arrays for efficient fault characterization in complex configuration spaces", Presented at the ACM SIGSOFT Software Engineering Notes, 2004.

[13] S. Maity, A. Nayak, M. Zaman, N. Bansal, A. Srivastava, "An improved test generation algorithm for pair-wise testing", International Symposium on Software Reliability Engineering (ISSRE), 2003.

[14] Y. Lei, R. Kacker, D.R. Kuhn, V. Okun and J. Lawrence, "IPOG/IPOG-D: efficient test generation for multi-way combinatorial testing", Software Testing, Verification & Reliability, vol. 18, pp. 125-148, 2008.

[15] A. Hartman, IBM Intelligent Test Case Handler, IBM alpha works, 2005, http://www.alphaworks.ibm.com/tech/whitch.

[16] K. Z. Zamli, M. F.J. Klaib, M. I. Younis,N. A. M. Isa, R. Abdullah, "Design and implementation of a t-way test data generation strategy with automated execution tool support", Information Sciences, vol. 181, pp. 1741–1758, 2011.

[17] S. Fouch, M.B. Cohen and A. Porter, "Towards incremental adaptive covering arrays", in: Presented at the 6th Joint Meeting on European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering: Companion Papers, Dubrovnik, Croatia, 2007.

[18] V. Rafe, "Scenario-driven analysis of systems specified through graph transformations", J. Visual Lang. Comput. 24 (2013) 136–145.

ISSN: 2745-3673
http://www.Rcsj.ir

[19] M. Woodside, G. Franks, D.C. Petriu, The future of software performance engineering, in: Future of Software Engineering Conference, FOSE '07, IEEE Computer Society, Minneapolis, MN, USA, 2007, pp. 171–187.
[20] D.M. Cohen, S.R. Dalal, M.L. Fredman and G.C. Patton, "The AETG system: an approach to testing based on combinatorial design", IEEE Transactions on Software Engineering vol. 23 pp. 437–444, 1997.
[21] J. Czerwonka, "Pairwise testing in real world: practical extensions to test case generator", in: 24th Pacific Northwest Software Quality Conference, IEEE Computer Society, Portland, OR, USA, pp. 419–430, 2006.
[22] R.C. Bryce, C.J. Colbourn, "The density algorithm for pairwise interaction testing: research articles", Software Testing, Verification & Reliability, vol. 17, pp.159–182, 2007.
[23] R.C. Bryce, C.J. Colbourn, "A density-based greedy algorithm for higher strength covering arrays", Software Testing, Verification & Reliability, vol. 19, pp. 37–53, 2009.
[24] Y.T. Yu, S.P. Ng and E.Y.K. Chan, "Generating, selecting and prioritizing test cases from specifications with tool support", in: 3rd International Conference on Quality Software, IEEE Computer Society, Dallas, TX, pp. 83–90, 2003.
[25] E. Lehmann, J. Wegener, "Test case design by means of the CTE XL", in: 8th European International Conference on Software Testing", Analysis & Review, pp. 1–10, 2000.
[26] J. Arshem, "TVG download page", 2009. http://sourceforge.net/projects/tvg.
[27] M.B. Cohen, Designing Test Suites for Software Interaction Testing, PhD Thesis Department of Computer Science, University of Auckland, New Zealand, 2004.
[28] M.B. Cohen, M.B. Dwyer, J. Shi, Interaction testing of highly-configurable systems in the presence of constraints, Proceeding of International Symposium on Software Testing and Analysis, ACM, London, UK, 2007, pp. 129–139.
[29] B. Jenkins, Jenny download web page, Bob Jenkins' Website, 2005. http://burtleburtle.net/bob/math/jenny.html.
[30] M.B. Cohen, "Designing Test Suites for Software Interaction Testing", Department of Computer Science, University of Auckland, p. 185, 2004.
[31] T. Shiba, T. Tsuchiya, T. Kikuno, Using artificial life techniques to generate testcases for combinatorial testing, in: 28th Annual International Computer Softwareand Applications Conference, vol. 71, IEEE Computer Society, Hong Kong , pp. 72–77, 2004.
[32] M.B. Cohen, P.B. Gibbons, W.B. Mugridge, C.J. Colbourn and J.S. Collofello, "Variable strength interaction testing of components", in: 27th Annual International Conference on Computer Software and Applications, IEEE Computer Society, Dallas, TX, USA, pp. 413–418, 2003.
[33] X. Chen, Q. Gu, A. Li and D. Chen, "Variable strength interaction testing with an ant colony system approach", in: 16th Asia-Pacific Software Engineering Conference, IEEE Computer Society, Penang, Malaysia, pp. 160–167, 2009.
[34] S. Margetts, "Adaptive Genotype to Phenotype Mappings for Evolutionary Algorithms", a dissertation submitted in partial fulfilment of the requirements of Cardiff University for the degree of Doctor of Philosophy, Department of Computer Science, Cardiff University, 2001.
[35] P. Flores, Y. Cheon, "PWiseGen: generating test cases for pairwise testing U sing genetic algorithms," IEEE International Conference on Computer Science and Automation Engineering (CSAE), 2011.
[36] J. McCaffrey, "An empirical study of pairwise test set generation using a genetic algorithm," Proceedings of the 7th International Conference on Information Technology, IEEE Computer Society, 2010, pp. 992–997.

ISSN: 2745-3673
http://www.Rcsj.ir

[37] S. Sabharwal, P. Bansal, N. Mittal, S. Malik, "Construction of mixed covering arrays for pair-wise testing using probabilistic approach in genetic algorithm," Arabian J. Sci. Eng. 41 (2016) 2821–2835.

[38] P. Bansal, S. Sabharwal, S. Malik, V. Arora, V. Kumar, An approach to test set generation for pair-wise testing using genetic algorithms, Proceedings of the 5th International Symposium on Search Based Software Engineering, 8084 2013, pp. 294–299.

[39] P. Bansal, S. Sabharwal, N. Mittal, S. Arora, Construction of variable strength covering array for combinatorial testing using a greedy approach to genetic algorithm, e-Inf. Softw. Eng. J. 9 (2015) 87–105.

[40] S. Esfandyari, V. Rafe, A tuned version of genetic algorithm for efficient test suite generation in interactive t-way testing strategy, Information and Software Technology 94 (2018) 165–185.

[41] K.Z. Zamli, B.Y. Alkazemib, G. Kendall, A Tabu Search hyper-heuristic strategy for tway test suite generation, Appl. Soft Comput. 44 (2016) 57–74.

[42] H. Wu, C. Nie, F. Kuo, H. Leung, C.J. Colbourn, A discrete particle swarm optimization for covering array generation, IEEE Trans. Evol. Comput. 19 (2015) 575–591.

[43] T. Mahmoud, B.S. Ahmed, An efficient strategy for covering array construction with fuzzy logic-based adaptive swarm optimization for software testing use, Expert Syst. Appl. 42 (2015) 8753–8765.

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

[۴۵] زهرا عباسی، سجاد اسفندیاری و وحید رافع، « ساخت آرایه پوشش با استفاده از الگوریتم بهینه‌سازی مبتنی برآموزش و یادگیری»، مجله مهندسی برق دانشگاه تبریز، انتشار آنلاین ۲۴ اردیبهشت ۱۳۹۷.

[46] Y. Lei, R. Kacker, D.R. Kuhn, V. Okun, J. Lawrence, IPOG: a general strategy for tway software testing, Proceedings of the 4th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, IEEE Computer Society, Tucson, AZ, 2007, pp. 549–556.

[47] M. Forbes, J. Lawrence, Y. Lei, R.N. Kacker, D.R. Kuhn, Refining the in parameterorder strategy for constructing covering arrays, J. Res. Nat. Inst. Stand. Technol. 113 (2008) 287–297.

[48] A. Calvagna, A. Gargantini, IPO-s: incremental generation of combinatorial interaction test data based on symmetries of covering arrays, IEEE International Conference on Software Testing, Verification, and Validation Workshops, Denver, CO, USA, IEEE Computer Society, 2009, pp. 10–18.

[49] Z. Wang, B. Xu, C. Nie, Greedy heuristic algorithms to generate variable strength combinatorial test suite, Proceedings of the 8th International Conference on Quality Software, Oxford, UK, IEEE Computer Society, 2008, pp. 155–160.

[1]Horizontal extension
[2]Pairwise (2-way) interactions.
[3]Ant Colony System
[4]Fuzzy Self Adaptive PSO
[5]efficiency
[6]performance
[7]Not Available
[8]Not Support
[9]Bold