ساخت آرایه پوشش با استفاده از الگوریتم بهینه‌سازی مبتنی برآموزش و یادگیری

زهرا عباسی[۱]، کارشناس ارشد؛ سجاد اسفندیاری[۲]، کارشناس ارشد؛ وحید رافع[۳]، دانشیار

۱- دانشکده فنی مهندسی - دانشگاه اراک - اراک - ایران – z-abbasi@msc.araku.ac.ir
۲- دانشکده فنی مهندسی - دانشگاه اراک - اراک - ایران – s-esfandyari@arshad.araku.ac.ir
۳- دانشکده فنی مهندسی - دانشگاه اراک - اراک – ایران – v-rafe@araku.ac.ir

چکیده: در سیستم‌های نرم‌افزاری اغلب خطاهای غیرمنتظره زمانی رخ می‌دهد که هم‌زمان تعدادی از اجزاء سیستم باهم در تعامل باشند. آزمون ترکیباتی روشی است که هدف آن تولید دنباله آزمون کمینه است تا خطاهایی که توسط این اجزاء و تعامل آن‌ها به وجود می‌آید را مشخص کند. تولید آرایه پوشش یک مسئله بهینه‌سازی است که یکی از پرطرفدارترین حوزه‌های پژوهش در زمینه آزمون‌های ترکیباتی است. الگوریتم‌های فرامکاشفه‌ای در تولید آرایه پوشش نتایج خوبی را داشته‌اند. گرچه این استراتژی‌ها نتایج بسیار خوبی دارند اما به دلیل پیچیدگی، مراحل جستجوی آن‌ها زمان‌بر است. به این دلیل این استراتژی‌ها به تولید آرایه پوشش برای پیکربندی‌های کوچک محدود هستند و قابلیت پشتیبانی تا قوه‌ی $t \leq 6$ را دارند. در این پژوهش ما با استفاده از الگوریتم بهینه‌سازی مبتنی بر آموزش و یادگیری و طراحی تابع برازندگی، به‌گونه‌ای سرعت جستجو را بالابرده‌ایم که الگوریتم توانایی تولید آرایه پوشش تا قوه $t = 15$ را دارد. علاوه‌براین، توانایی الگوریتم در کمینه‌سازی دنباله آزمون نیز بالا است. نتایج نشان می‌دهد که الگوریتم پیشنهادی قادر به تولید نتایج بسیار بهتری نسبت به سایر روش‌های موجود است.

واژه‌های کلیدی: آزمون مبتنی بر ترکیبات، آرایه پوشش، الگوریتم بهینه‌سازی مبتنی بر آموزش و یادگیری.

Covering Array Generation using Teaching Learning base Optimization Algorithm

Z. Abbasi[1], MSc; S. Esfandyari[2], MSc; V. Rafe[3], Associate Professor

1- Faculty of Engineering, University of Arak, Arak, Iran, Email: z-abbasi@msc.araku.ac.ir
2 Faculty of Engineering, University of Arak, Arak, Iran, Email: s-esfandyari@arshad.araku.ac.ir
3- Faculty of Engineering, University of Arak, Arak, Iran, Email: v-rafe@araku.ac.ir

Abstract: In software systems, the most of unexpected bugs usually occur when some of the system components interact with each other at the same time. Combinatorial testing is a method that aims to generate minimized test suite to determine the bugs caused by these components and their interactions. Covering array generation, an optimization problem, is the most popular research area in the field of combinatorial testing. Meta-heuristic algorithms have succeeded in generating covering arrays. Although good results can be found by these strategies, their complex search processes are time consuming. In this respect, these strategies have been confined to small configurations and they can support small interaction strengths $(t \leq 6)$. In this research, teaching learning base optimization algorithm has been used and fitness function has been designed in a way that searching process become so fast that our strategy enabled to generate covering array with interaction strength up to $t = 15$. In addition, ability of the algorithm in minimization of test suite is also high. The results of this current study showed that our proposed algorithm is able to generate the results more appropriate than the existing strategies.

Keywords: Combinatorial testing, Covering array, Teaching learning base optimization algorithm.

تاریخ ارسال مقاله: ۱۳۹۵/۰۸/۱۶
تاریخ اصلاح مقاله: ۱۳۹۵/۱۰/۲۲ و ۱۳۹۵/۱۲/۰۶
تاریخ پذیرش مقاله: ۱۳۹۵/۱۲/۱۵
نام نویسنده مسئول: وحید رافع
نشانی نویسنده مسئول: ایران - اراک - سردشت - پردیس - دانشگاه اراک - دانشکده فنی و مهندسی - گروه کامپیوتر

۱- مقدمه

نرم‌افزارهای کاربردی مدرن اغلب پیچیده و دارای اجزای متعددی هستند به این دلیل آزمون سیستم‌های نرم‌افزاری، فرایندی هزینه‌بر شده است. یک مسئله کلیدی در حوزه تولید نمونه آزمون[۱]، کشف خطا با هزینه کم است. آزمون ترکیباتی[۲] یک روش پرطرفدار برای کشف خطاهایی است که توسط اجزای مختلف سیستم و تعامل بین آن‌ها رخ می‌دهند [۱]. تمام تعاملات ممکن از اجزای سیستم نرم‌افزاری شامل فضای بزرگی است. آرایه پوشش[۳] به‌عنوان دنباله آزمون در آزمون ترکیباتی، با نمونه‌برداری تعداد کمی از این تعاملات متفاوت از سیستم که همان نمونه آزمون‌ها هستند تشکیل می‌شود؛ این نمونه‌برداری باید به‌گونه‌ای انجام شود که تمام ترکیبات ممکن بین تعداد مشخصی از پارامترها را پوشش دهد. Kuhn و Reilly نشان می‌دهند [۲] که بیش از ۷۰٪ از خطاها در برخی از نرم‌افزارها با تعامل بین یک یا دو پارامتر ایجاد می‌شوند و تقریباً تمام خطاها با بررسی تعاملات بین شش پارامتر می‌توانند شناسایی شوند. درنتیجه آزمون ترکیباتی یک روش مؤثر در حوزه آزمون نرم‌افزار است.

تولید آرایه پوشش کمینه یک چالش عمده در آزمون ترکیباتی است. استراتژی t-ستونی[۴] بر تولید بهینه آرایه پوشش که یک مسئله بهینه‌سازی NP-سخت[۵] است [۵-۳] تمرکز دارد. هدف اصلی، تولید آرایه پوشش کمینه با زمان جستجوی کم و هزینه مناسب است. پژوهش‌های زیادی برای حل این مسئله وجود دارند که به دو گروه اصلی تقسیم می‌شوند:

۱) روش‌های ریاضی[۶] [۱، ۵، ۶]
۲) روش‌های محاسباتی[۷] [۱، ۷].

روش‌های ریاضی از برخی از ساختارهای ترکیباتی مربوط به توابع ریاضی مانند آرایه متعامد[۸] استنتاج شده‌اند اما روش‌های محاسباتی عمدتاً از استراتژی‌های حریصانه یا تکنیک‌های فرامکاشفه‌ای برای تولید آرایه پوشش بهینه در فضای جستجو استفاده می‌کنند. روش‌های ریاضی آرایه پوشش بهینه را فقط برای پیکربندی‌های خاصی تولید می‌کنند [۸] از استراتژی‌های موجود در روش‌های ریاضی به Combinatorial Test Services و (TConfig) [۹] که بر اساس تعمیم آرایه متعامد ساخته شده‌اند می‌توان اشاره کرد. عمده‌ترین مشکل راهکارهای مبتنی بر آرایه متعامد این است که توانایی ساخت آرایه پوشش برای پیکربندی‌های کوچک و خاص را دارند که این محدودیت باعث استفاده از روش‌های محاسباتی شده است.

راهکار اغلب روش‌های محاسباتی به این صورت است که در ابتدا تمام ترکیبات ممکن با توجه به مشخصات ورودی تولید می‌شوند پس از آن نمونه آزمون‌ها برای پوشش این ترکیبات ساخته می‌شوند. تفاوت اصلی بین استراتژی‌های مختلف روش‌های محاسباتی در ساختن نمونه آزمون است. دو روش اصلی برای ساخت نمونه آزمون در روش‌های محاسباتی وجود دارد[۱، ۷، ۸]:

۱) یک-آزمون-در-یک-زمان[۹]
۲) یک-پارامتر-در-یک-زمان[۱۰]

در روش یک-آزمون-در-یک-زمان در هر مرحله یک نمونه آزمون یا یک مجموعه کامل از نمونه آزمون‌ها تولید می‌شوند سپس نمونه آزمونی که بیش‌ترین ترکیبات را پوشش دهد انتخاب می‌شود، این نمونه آزمون یک سطر از دنباله آزمون نهایی (آرایه پوشش) را تشکیل می‌دهد. معروف‌ترین استراتژی‌هایی بر اساس این روش AETG [۱۰]، mAETG [۱۱]، PICT [۱۲]، DDA [۱۳، ۱۴]، CTE-XL [۱۵، ۱۶]، TVG [۱۷، ۱۸]، Jenny [۱۹]، GTWay [۲۰] و ITCH [۲۱] است. روش یک-پارامتر-در-یک-زمان آرایه پوشش را با اضافه کردن هر پارامتر مرحله به مرحله می‌سازد و سپس با اضافه کردن ترکیبات ممکن برای پارامترهای موجود مراحل را تا زمانی که پوشش کامل شود ادامه می‌دهد. IPOG [۲۲] ، IPOG-s [۲۳] و IPOG-D [۸] استراتژی‌های رایجی هستند که بر اساس این روش گسترش داده شده‌اند.

دسته‌ی دیگری از روش‌های مبتنی بر محاسبات الگوریتم‌های مبتنی بر هوش مصنوعی هستند. این الگوریتم‌ها بر اساس روش یک-آزمون-در-یک-زمان گسترش یافته‌اند. با کاربرد روز افزون مهندسی نرم‌افزار مبتنی بر جستجو[۱۱] در حل بهینه مسائل نرم‌افزار، استفاده از الگوریتم‌های فرامکاشفه‌ای در تولید آرایه پوشش از سایر روش‌ها پرطرفدارتر است. الگوریتم‌های فرا مکاشفه‌ای با یک مجموعه تصادفی از راه‌حل‌ها شروع می‌شوند سپس به‌منظور بهبود این راه‌حل‌های اولیه یک سری تغییرات بر روی آن‌ها انجام می‌شوند و در آخر بهترین راه‌حل انتخاب می‌شود این مراحل تا زمانی که تمام ترکیبات پوشش داده شوند ادامه می‌یابند. تعدادی از الگوریتم‌های فرامکاشفه‌ای که تاکنون استفاده شده‌اند SA [۲۴]، TS [۲۵]، GA [۲۶]، ACA [۲۶، ۲۷]، PSO [۳۰-۲۸]، CS [۳۱]، HSS [۳۲]، H-B [۳۳]، BA [۳۴] و BPTS [۳۵] هستند.

الگوریتم SA نتایج بهینه‌ای را برای پیکربندی‌های کوچک ($t < ۳$) تولید می‌کند. الگوریتم‌های GA و TS نیز وقتی ابعاد مسئله بالا برود ($t > ۳$) قادر به تولید جواب نیستند. الگوریتم H-B تا $t = ۱۰$ را پشتیبانی می‌کند اما به دلیل زمان‌بر بودن فرآیند جستجو فقط برای آرایه‌های پوشش محدودی توانایی تولید دنباله آزمون را دارد. الگوریتم CS و PSO توانایی پشتیبانی تا $t = ۶$ را دارند. الگوریتم PSO مشکلاتی از قبیل تنظیم کردن پارامتر و همگرایی زودرس که باعث گیر افتادن در بهینه محلی می‌شود، را دارد که این عوامل تأثیر بسزایی بر توانایی بهینه‌سازی آن می‌گذارند. مشکل تمام این استراتژی‌ها محاسبات سنگین آن‌ها است و این‌که در تولید دنباله آزمون نهایی از دقت بالایی برخوردار نیستند. برای مثال الگوریتم ژنتیک از پیچیدگی مراحل ادغام و جهش رنج می‌برد. ACA هنگامی که تعداد مورچه‌ها زیاد می‌شوند با مشکلات متعددی روبه‌رو می‌شود. TS از مکانیسم به‌روزرسانی لیست ممنوعه رنج می‌برد. علاوه‌براین، استراتژی‌های موجود با این مسئله مواجه هستند که باید یک توازنی بین دقت الگوریتم و سرعت آن برقرار کنند [۳۶].

دو مشکل عمده‌ای که در کارهای انجام شده وجود دارد: اول، پیچیدگی ساختار الگوریتم‌ها و فرایند جستجو و دوم، وابستگی الگوریتم‌ها به تنظیم پارامترهای متعدد که باعث محدودیت کارایی آن‌ها شده است. بنابراین، باتوجه به اهمیت راهکارهای فرامکاشفه در حل مسئله پوشش آرایه این دو مشکل باید برطرف شود. در این مقاله از الگوریتم بهینه‌سازی مبتنی بر آموزش و یادگیری[۱۲] (TLBO) استفاده شده است که یکی از الگوریتم‌های بهینه‌سازی کارآمد و جدید است. از مزایای این الگوریتم می‌توان به ساختار ساده آن، توانایی جستجوی بالا و تنوع‌پذیری در تولید جواب اشاره کرد. برای سرعت بخشیدن به فرآیند جستجو و سادگی محاسبات ساختمان داده‌ای برای ذخیره ترکیبات طراحی شده است که فضای مورد نیاز برای ذخیره‌سازی را بسیار کاهش داده است و به همین دلیل سرعت الگوریتم به‌طور چشم‌گیری بالا رفته است به‌طوری که توانایی پشتیبانی از پیکربندی‌ها در ابعاد بزرگ تا t ۱۵ = را دارد.

مطالبی که در ادامه مقاله می‌آید به این صورت است: در بخش دوم، مروری بر مفاهیم آزمون‌های ترکیباتی، آرایه پوشش و الگوریتم TLBO بخش سوم، معرفی راهکار پیشنهادی، به‌کارگیری الگوریتم TLBO در ساخت آرایه پوشش و نحوه طراحی ساختمان داده برای ذخیره ترکیبات بخش چهارم، بررسی کارایی الگوریتم پیشنهادی در مقایسه با کارهای موجود بخش پنجم، نتیجه‌گیری مقاله و کارهای آینده آورده شده است.

۲- تعاریف و مفاهیم پایه

۲-۱- آزمون ترکیباتی

فرض کنید رفتار نرم‌افزاری که آزمون می‌شود توسط k پارامتر مستقل کنترل می‌شود که می‌توانند نشان‌دهنده رویدادهای داخلی و خارجی نرم‌افزار، ورودی که کاربر وارد می‌کند و یا حالت‌هایی از تنظیم پارامترهای سیستم باشند. هر پارامتر i، v_i تا مقدار گسسته را می‌تواند داشته باشد. این مقادیر از مجموعه‌ی متناهی M_i که تعداد اعضای آن v_i است، انتخاب می‌شود. هر k تایی $(x_1, x_2, x_3,\ldots,x_k)$ یک نمونه آزمون را که $x_i \in M_i$ و $1 \le i \le k$ نشان می‌دهد. یک مثال فرضی از تلفن همراه را در نظر بگیرید. فرض کنید این سیستم از چهار اجزا یا پارامتر مستقل تشکیل شده است. که عبارت است از: نوع تماس، نوع پیام، رسانه و صفحه نمایش که هرکدام از این پارامترها سه مقدار ممکن را می‌توانند بگیرند. جدول ۱ این پارامترها و مقادیرشان را نشان می‌دهد. در این مثال $k = 4$ و $v_1= v_2= v_3= v_4=3$ است. چهارتایی (کیفیت بالا، رادیو، عکس، صوتی) یک نمونه آزمون از این سیستم را تشکیل می‌دهد.

خطای نرم‌افزار زمانی که نرم‌افزار در حالت خاصی تنظیم شود به وجود می‌آید در مثال تلفن همراه با مقداردهی هر پارامتر یک تنظیم خاص در نرم‌افزار به وجود می‌آید که ممکن است در یکی از این تنظیمات تعامل همزمان چندین پارامتر باهم باعث ایجاد خطا در عملکرد نرم‌افزار شود. بنابراین، برای شناسایی خطا و آزمون نرم‌افزار باید تمام تنظیمات متفاوت از نرم‌افزار تولید شوند. همان‌طور که گفته شد، هر تنظیمات خاص از نرم‌افزار توسط یک نمونه آزمون مشخص می‌شود. مجموعه‌ای از این نمونه آزمون‌ها یک دنباله آزمون را تشکیل می‌دهند. برای شناسایی خطا باید دنباله آزمون، تمام نمونه آزمون‌های ممکن را شامل شود.

جدول ۱: پیکربندی تلفن همراه [۳۷]

صفحه نمایش	رسانه	پیام	تماس
سیاه و سفید	دوربین	متن	صوتی
رنگ پایه	رادیو	عکس	تصویری
کیفیت بالا	ویدئو	ویدئو	صوتی و تصویری

برای مثال وقتی پارامتر نوع تماس، صوتی است و صفحه نمایش بر روی، کیفیت بالا تنظیم است خطای سیستم رخ می‌دهد. برای شناسایی این خطا حالت دوتایی (کیفیت بالا، -، -، صوتی) حداقل یک‌بار باید توسط دنباله آزمون پوشش داده شود. برای ساده کردن کار از اندیس به‌جای مقادیری که هر پارامتر می‌تواند بگیرد استفاده می‌شود. به‌عنوان مثال از (۲، -، -، ۰) به‌جای (کیفیت بالا، -، -، صوتی) استفاده شده است.

در آزمون کامل[۱۳] نرم‌افزار تمام ترکیبات مختلف ممکن از پارامترهای سیستم باید پوشش داده شوند. در مثال تلفن همراه نیاز به تولید ۸۱=۳×۳×۳×۳ نمونه آزمون است. هزینه آزمون زمانی که تعداد پارامترها و تعداد مقادیر آن‌ها زیاد باشد، بسیار بالا می‌رود. علاوه‌براین، چون تعامل تعداد محدودی از پارامترها باعث بروز خطا می‌شود [۲]، تولید آزمون کامل اصلاً مقرون‌به‌صرفه نیست و باید از تکنیکی دیگر برای حل این مسئله استفاده کرد. بنابرین، به‌جای آزمون کامل از آزمون t-ستونی استفاده می‌کنیم که مقدار t در بازه $2 \le t \le k$ می‌باشد. مقدار t مشخص کننده این است که دنباله آزمون ما به‌جای این‌که تمام ترکیبات متفاوت از n پارامتر سیستم را پوشش دهد فقط شامل ترکیبات t پارامتر از سیستم 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 نمی‌توان شناسایی کرد.

آزمون t-ستونی تمام ترکیبات t-تایی ممکن از پارامترها را پوشش می‌دهد. دنباله آزمونی که در این آزمون تولید می‌شود، آرایه پوشش نامیده می‌شود. t قوه پوشش است و مقدار آن عمق پوشش را مشخص می‌کند. t یک تنظیم کلیدی برای آزمون t-ستونی است که باید توسط آزمون کننده مشخص شود. در ادامه تعریف دقیق‌تری از آرایه پوشش را بیان می‌کنیم. به‌طورکلی آزمون t-ستونی با مفهوم آرایه پوشش در ریاضیات ارتباط مستقیمی دارد به این دلیل از مفهوم آرایه پوشش برای تعریف آزمون t-ستونی استفاده می‌شود [۳۸]. یک آرایه N×K که تعداد نمونه آزمون‌های آن N است دارای دو ویژگی زیر است:

۱- هر ستون i که $1 \le i \le k$ شامل عضوهایی از مجموعه M_i باشد که $|M_i|=v_i$

۲- تعداد سطرهای هر زیر آرایه N×t از آن، تمام ترکیبات $|M_{k1}|\times|M_{k2}|\times..\times|M_{kt}|$ از t ستون را حداقل یک‌بار پوشش دهد.

آرایه پوشش با نماد $CA(N; t, k, v_1, v_2, v_3, \ldots, v_k)$ و درصورتی که $v_1 = v_2 = \ldots = v_k = v$ با نماد $CA(N; t, k, v)$ و یا $CA(N; t, k)$ نمایش داده می‌شود.

برای ساخت آرایه پوشش در مثال تلفن همراه که تمام تعاملات دو تایی از پارامترها را پوشش دهد، تنها نیاز به ۹ نمونه آزمون است که در جدول ۲ نشان داده شده است. در آزمون دو ستونی فقط تعاملات دوتایی از پارامترها در نظر گرفته می‌شوند. درصورتی که آزمون کامل، شامل تمام تعاملات است و برای مثال تلفن همراه، نیاز به ۸۱ نمونه آزمون دارد. جدول ۲ آرایه پوشش یا دنباله آزمون دو ستونی را نشان می‌دهد که هر سطر آن یک نمونه آزمون است. در جدول ۲ تمام ترکیبات ممکن از هر دو پارامتر سیستم، موجود است برای مثال دو پارامتر نوع تماس و رسانه، دارای ۹=۳×۳ حالت متفاوت دوتایی هستند. که عبارت‌اند از: (-،دوربین ، -، صوتی)، (-،رادیو، -، صوتی)، (-، ویدئو، -، صوتی)، (-،دوربین ، -،تصویری)، (-،رادیو، -،تصویری)، (-، ویدئو، -،تصویری)، (-،دوربین ، -،صوتی‌تصویری)، (-، رادیو، -،صوتی‌تصویری)، (-، ویدئو، -، صوتی‌تصویری) که تمام این حالت‌ها را در جدول می‌توان مشاهده کرد.

جدول ۲: دنباله آزمون دو-ستونی برای تلفن همراه [۳۷]

شماره نمونه آزمون	صفحه نمایش	رسانه	پیام	تماس
۱	سیاه و سفید	ویدئو	عکس	صوتی
۲	رنگ پایه	ویدئو	ویدئو	صوتی تصویری
۳	رنگ پایه	رادیو	متن	صوتی
۴	کیفیت بالا	دوربین	ویدئو	صوتی
۵	سیاه و سفید	دوربین	متن	صوتی تصویری
۶	کیفیت بالا	ویدئو	متن	تصویری
۷	کیفیت بالا	رادیو	عکس	صوتی تصویری
۸	رنگ پایه	دوربین	عکس	تصویری
۹	سیاه و سفید	رادیو	ویدئو	تصویری

در اغلب سیستم‌های نرم‌افزاری اهمیت تعامل بین پارامترها ثابت نیست برخی از تعاملات بیش‌تر مستعد خطا هستند درحالی که برخی از تعاملات دیگر تأثیر کم‌تری بر خطای سیستم دارند. برای این‌که بتوانیم این تعاملات متفاوت را به‌طور مؤثر شناسایی کنیم از ساختار آرایه پوشش با قوه متغیر[۱۴] 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 $VSCA(N; t, k, v, \{C\})$ نشان داده می‌شود. {C} یک مجموعه از آرایه‌های پوشش است که می‌تواند شامل یک یا چند $CA(t', k', v)$ باشد که k' باید زیر مجموعه‌ای از k باشد و t' بزرگتر از t باشد.

۲-۲- الگوریتم TLBO

الگوریتم بهینه‌سازی مبتنی بر آموزش و یادگیری (TLBO) در سال ۲۰۱۱ معرفی شد [۳۹]. TLBO الگوریتمی برگرفته از فرآیند آموزش و یادگیری است. که بر مبنای تأثیر آموزگار بر بازده دانش‌آموزان در یک کلاس بنا نهاده شده است. این الگوریتم مانند سایر الگوریتم‌های فرامکاشفه‌ای دارای توانایی جستجوی سراسری بالایی است. همچنین به‌دلیل ویژگی نزدیک شدن به سایر دانش‌آموزان (فاز دانش‌آموز) تنوع‌پذیری در الگوریتم بالا رفته است و مانند برخی الگوریتم‌ها مانند الگوریتم بهینه‌سازی توده ذرات فقط به سمت بهترین‌ها حرکت نمی‌کند و این عامل باعث می‌شود از همگرایی زودرس جلوگیری شود.

در تمام الگوریتم‌های مبتنی بر هوش مصنوعی تنظیم پارامترهای اندازه جمعیت و تعداد تکرار لازم است که پارامترهای عمومی نامیده می‌شوند . علاوه‌براین پارامترهای عمومی هر الگوریتم نیز یک سری پارامترهای اختصاصی دارد. تنظیم این پارامترهای اختصاصی عملکرد الگوریتم را تحت تأثیر قرار می‌دهند. تنظیم نادرست این پارامترها محاسبات را افزایش می‌دهند و یا جواب بهینه محلی تولید می‌کنند. الگوریتم بهینه‌سازی مبتنی بر آموزش و یادگیری (TLBO) از جدیدترین الگوریتم‌های فرامکاشفه‌ای است که پارامترهای اختصاصی ندارد و این خود باعث سهولت و کارایی آن می‌شود.

الگوریتم TLBO یک الگوریتم جمعیت‌گرا است که از جمعیتی از راه‌حل‌ها برای رسیدن به راه‌حل بهینه استفاده می‌کند. در الگوریتم TLBO این جمعیت به‌عنوان گروهی از دانش‌آموزان در نظر گرفته می‌شوند. استاد بهترین راه‌حل به‌دست‌آمده تاکنون می‌باشد. فرآیند الگوریتم TLBO به دو قسمت تقسیم می‌شود: اولین قسمت، فاز استاد نام دارد که به معنای یادگیری دانش‌آموزان از استاد است، قسمت دوم، فاز دانش‌آموز است که به معنای تعامل بین دانش‌آموزها برای یادگیری بهتر است.

۲-۲-۱- فاز استاد

استاد سعی می‌کند با آموزش به دانش‌آموزان سطح علمی آن‌ها را بالا ببرد. که این کار را با استفاده از یک رابطه تصادفی مطابق با فرمول(۱) انجام می‌دهد.

$$X_{new,D} = X_{old,D} + r(X_{teacher,D} - T_F M_D) \quad (۱)$$

اندیس D تعداد دروس (متغیر مسئله)، $X_{old,D}$ عضو قدیمی است که یک بردار D بعدی است. r یک عدد تصادفی در بازه [۰ و ۱]، $X_{teacher,D}$ بهترین عضو جمعیت است T_F پارامتر تدریس یک متغیر تصادفی است که مقدار آن ۱یا ۲ می‌باشد و M_D میانگین کلاس است. عضو جدید $X_{new,D}$ درصورتی که بهتر از عضو قدیمی باشد پذیرفته می‌شود.

۲-۲-۲- فاز دانش‌آموز

هر دانش‌آموز به‌منظور افزایش دانش خود با دانش‌آموزان دیگر تعامل برقرار می‌کند. به این صورت که هر دانش‌آموز به‌صورت تصادفی با دیگر دانش‌آموزان جهت افزایش کارایی تعاملاتی مثل مباحثه، ارائه و سؤال را انجام می‌دهد و مفاهیم جدیدی را از دانش‌آموزانی که دانش بیش‌تری دارند یاد می‌گیرد.

$$X_{new,i} = X_{old,i} + r_i(X_j - X_K) \quad (۲)$$

اندیس i از یک تا تعداد کل اعضا تغییر می‌کند. $X_{old,\ i}$ مقدار قدیمی دانش‌آموز i است. r_i یک عدد تصادفی در بازه [۱ و ۰] است. X_k وX_j دو دانش‌آموز هستند که به‌صورت تصادفی با شرط j≠k و بهتر بودن تابع برازندگی X_j نسبت به X_k انتخاب می‌شوند. عضو جدید $X_{old,i}$ درصورتی که بهتر از عضو قدیمی باشد پذیرفته می‌شود.

۳- راهکار پیشنهادی

در این بخش نحوه کاربرد الگوریتم TLBO در راهکار جدید برای تولید آرایه پوشش شرح داده می‌شود. در ابتدا الگوریتم TLBO تعدادی نمونه آزمون که همان دانش‌آموزان هستند را تولید می‌کند. سپس تابع برازندگی بهترین نمونه آزمون (دانش‌آموز) را انتخاب می‌کند. درواقع تابع برازندگی تعداد ترکیباتی است که هر نمونه آزمون می‌تواند پوشش دهد. بنابراین، برای محاسبه برازندگی هر نمونه آزمون ابتدا باید تمام ترکیبات تولید شود و سپس مکانیسمی برای محاسبه تابع برازندگی از ترکیبات تولیدشده طراحی شود. درنتیجه الگوریتم از دو مرحله اصلی تشکیل شده‌است:

۱-تولید ماتریس پوشش (تولید کل عناصر t-تایی)

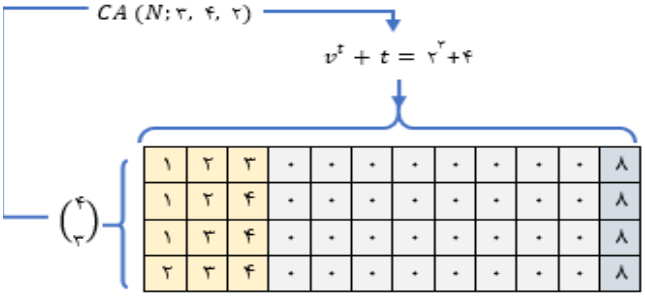
۲-تولید دنباله آزمون با الگوریتم TLBO

۳-۱- تولید ماتریس پوشش (CM)

مهم‌ترین نوآوری این پژوهش، در تولید بهینه ماتریس پوشش (CM) است. در درجه اول طراحی ماتریس پوشش به‌گونه‌ای انجام گرفته که حافظه مورد نیاز برای ذخیره‌سازی را به‌طور چشم‌گیری کاهش داده است. علاوه‌براین، مکانیسم جستجوی عناصر t-تایی موجود در ماتریس طوری طراحی شده است که این مرحله در مقایسه با روش‌های فرامکاشفه‌ای دیگر سرعت بسیار بالایی دارد و به همین دلیل توانایی پشتیبانی تا قوه t = ۱۵ و پیکربندی‌های با ابعاد بزرگ را دارد. برای ذخیره‌سازی حالات پوشش داده‌نشده (حالت اولیه) از ساختار بیتی استفاده می‌کنیم. منظور از ساختار بیتی این است که برای هر نمونه آزمون پوشش داده‌نشده یک بیت در نظر گرفته می‌شود. در این روش برخلاف روش‌هایی مانند CS و PSTG که از قوی‌ترین روش‌ها در تولید آرایه پوشش هستند، برای v^t نمونه آزمون یک سطر در نظر گرفته می‌شود و به تعداد ترکیبات C(k, t) سطر ایجاد می‌گردد. ما در این راهکار هر ترکیب را در سطر مربوطه ذخیره می‌کنیم. این کار ما را از سویچ کردن بین ماتریس‌ها بی‌نیاز می‌کند و با داشتن اعداد مربوط به هر ترکیب به‌راحتی می‌توان به سلول مورد نظر دسترسی پیدا کرد. به‌عنوان‌مثال برای CA (N; t, k, v) نیاز به C(k, t) سطرv^t+t+۱ و ستون است. برای نشان دادن اختلاف بین ساختمان داده اولیه، مثال CA(N;۲,۴,۲) را برای راهکار پیشنهادی و CS مقایسه می‌کنیم. CS به ماتریسی شامل C(۴,۲)×$۲^۲$ سطر و ۴ ستون نیاز دارد، که در مجموع شامل ۴×۲۴ سلول است و روش پیشنهادی به C(۴,۲) سطر و ($۲^۲$+۳) ستون و در مجموع ۷×۶ سلول (حتی با فرض در نظر گرفتن بایت به‌جای بیت) نیاز دارد، که این اختلاف برای پیکربندی‌های بزرگ، بسیار بیش‌تر است.

الگوریتم در ابتدا مشخصات آرایه پوشش $CA(N;t,k,v)$ را دریافت می‌کند. سپس یک ماتریس با ابعاد $C(k,t)\times(v^t+t)$ می‌سازد که سطرهای این ماتریس C(k,t) است و نشان‌دهنده تعداد ترکیبات t-تایی ممکن در بین k پارامتر است. در t ستون اول ماتریس، اندیس‌های ترکیبات مربوط به هر سطر را ذخیره می‌کنیم. v^t ستون بعد نشان‌دهنده مقادیر ممکن دهدهی معادل ترکیبات t-تایی است. تا این مرحله کل عناصر t-تایی که باید توسط الگوریتم پوشش داده شود، تولید شده است و درایه‌های مربوط به آن‌ها در ماتریس پوشش (CM) مقدار صفر می‌گیرند که به معنای عدم پوشش این عناصر است.

شکل ۱ ماتریس پوشش را برای ساختار CA(N;۳,۴,۲) نشان می‌دهد. تعداد سطرهای ماتریس برابر با شش C(۴،۲) و درایه‌های سه ستون اول، اندیس‌های متفاوت این چهار ترکیب است. در سطر اول، ستون اول، دوم و سوم اندیس (۱، ۲ و۳) بیان‌گر ترکیب پارامتر اول، پارامتر دوم و پارامتر سوم است و هشت ستون بعدی مقادیر دهدهی ممکن از مقادیر متناظر در این سه پارامتر است. برای مثال پارامتر اول، دوم و سوم هشت مقدار متفاوت می‌توانند داشته باشند. ۰۰۰ که معادل دهدهی آن صفر است (ستون چهارم)، ۰۰۱ معادل دهدهی آن یک است (ستون پنجم)، ۰۱۰ ستون که معادل دهدهی آن ۲ است. همان‌طور که ملاحظه می‌شود درایه‌های مربوط به این ستون‌ها صفر است، که به معنای عدم پوشش این عناصر است. سطرهای بعد نیز به همین ترتیب است. ستون آخر نیز مجموع عناصر پوشش‌داده‌نشده را در خود جای می‌دهد.



شکل ۱: ساختار ماتریس پوشش برای CA(N;۳,۴,۲)

۳-۲- تولید دنباله آزمون

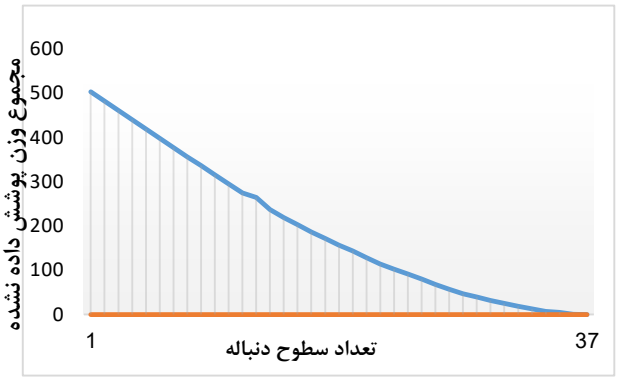
در الگوریتم پیشنهادی مرحله بعد از تولید ماتریس پوشش، انتخاب نمونه آزمون بهینه است. الگوریتم پیشنهادی با کمی تغییر در الگوریتم TLBO که از [۴۰] الهام گرفته است پیاده‌سازی شده است. مراحل پیاده‌سازی الگوریتم به این شرح است: همان‌طور که در شکل ۲ نشان داده شده است، حلقه While شرط خاتمه‌اش پوشش تمامی عناصر CM است. مقدار معلم در ابتدای هر مرحله به‌صورت تصادفی انتخاب می‌شود. اولین حلقه For به تعداد جمعیت دانش‌آموزان تکرار می‌شود که بسته به ابعاد آرایه پوشش تعداد جمعیت از ۳۰ تا ۲۵۰ متغیر است، که در بخش ۳-۳ به آن پرداخته می‌شود. این حلقه مربوط به فاز استاد است، که در این فاز طبق معادله (۱) تغییرات بر روی هر دانش‌آموز صورت می‌گیرد و در

انتها مقدار برازندگی آن با معلم مقایسه شده و در صورت برتری مقدارش جایگزین معلم می‌شود. حلقه For دوم مربوط به فاز دانش‌آموز است که نیمی از عناصر هر دانش‌آموز به‌صورت تصادفی انتخاب و یک مقدار تصادفی می‌گیرند و درصورت بهبود، این مقدار جایگزین مقدار قبل دانش‌آموز می‌شود و در انتها اگر مقدار دانش‌آموز بهتر از مقدار معلم شود این مقدار جایگزین مقدار معلم می‌شود. این حلقه به تعداد حداکثر ۵ و ۵٪ از تعداد جمعیت تکرار می‌شود. این مراحل برای تمام دانش‌آموزان کلاس انجام می‌شوند و در پایان حلقه مقدار معلم به دنباله آزمون TS اضافه می‌شود. در پایان دنباله آزمون نهایی، خروجی الگوریتم می‌شود.

```
1: Input (t,k,v)
2: Output: test suite
3: teacher=NULL
4: while CM ≠ Ø do
5:         initialize the value of teacher randomly
6:         for i=1 to population size do
7:             initialize the value of  learner randomly
8:             compute the fitness value fitness(li)
9:             if fitness(li) > fitness(teacher) then
10:                teacher = li
11:            end if
12:            update the learner according to the equation 1
13:            apply bound handling limitation
14:            compute the fitness value fitness(li)
15:            if fitness(li) > fitness(teacher) then
16:                teacher = li
17:            end if
18:        end for
19:        for i=1 to max(5, 5% of popsize)
16:        randomly reinitialize half of the learner value
17:            if fitness(li) > fitness(teacher) then
16:                teacher = li
17:            end if
18:        end for
19:        Add teacher to the test suite TS
20:    Remove the t-way schemas covered by teacher from CM
21: end while
22: return TS
```

شکل ۲: شبه‌کد تولید دنباله آزمون توسط الگوریتم TLBO

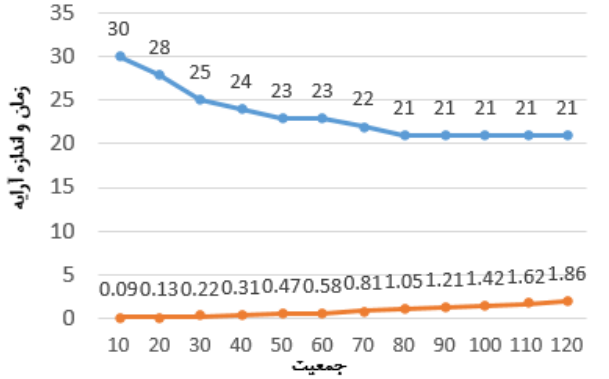
تابع برازندگی در الگوریتم پیشنهادی وزن هر نمونه آزمون را محاسبه می‌کند. تعریف وزن در تولید دنباله آزمون، تعداد حالات پوشش‌داده‌شده جدید است. درواقع در هر تکرار بهترین نمونه آزمون (نمونه آزمون با بیش‌ترین وزن) انتخاب می‌شود و پس از آن ماتریس پوشش به‌روز می‌شود و مقدار برازندگی یا وزن نمونه آزمون از وزن کل کم می‌شود، پس از آن نمونه آزمون به دنباله آزمون نهایی اضافه می‌شود. شکل ۳ منحنی تغییرات تابع برازندگی برای پیکربندی CA(N;۲,۷,۵) را نشان می‌دهد. در این پیکربندی تعداد کل حالات پوشش داده‌نشده اولیه ۵۰۴ است که پس از ۳۷ تکرار این تعداد به صفر می‌رسد و الگوریتم خاتمه می‌یابد. پس می‌توان نتیجه گرفت که تعداد سطرهای دنباله آزمون که الگوریتم TLBO برای پیکربندی CA(N;۲,۷,۵) تولید می‌کند، ۳۷ نمونه آزمون است.



شکل ۳: منحنی تغییرات تابع برازندگی

۳-۳- تعیین پارامتر الگوریتم TLBO

مهم‌ترین پارامتر در الگوریتم TLBO برای استراتژی پیشنهادی اندازه جمعیت افراد (popsize) است. افزایش جمعیت باعث بهبود نتیجه از لحاظ اندازه آرایه می‌شود اما باعث افزایش زمان تولید می‌شود. نکته قابل توجه این است که تعیین مقدار برای جمعیت در پیکربندی‌ها مختلف، متفاوت است. به‌عبارتی دیگر تغییر مقادیر k، t و مخصوصاً v در تعیین تعداد دفعات تکرار تأثیرگذار است. نتایج پیکربندی CA(N;۲,۴,۶) برای جمعیت افراد بین ۱۰ تا ۱۲۰ از لحاظ زمان و اندازه آرایه در شکل ۴ نشان داده شده است.



شکل ۴: زمان و اندازه آرایه برای CA(N;۲,۴,۶)

در حالت کلی با افزایش جمعیت افراد، دقت استراتژی افزایش می‌یابد. بدیهی است که استراتژی با جمعیت کم قادر به تولید دنباله آزمون بهینه نخواهد بود. شکل ۴ بهترین زمان و اندازه آرایه در ۲۰ بار آزمایش است. همان‌طوری که در این شکل مشاهده می‌شود، کم‌ترین زمان با جمعیت ۱۰ است که ۰/۰۹ ثانیه و بیش‌ترین زمان با جمعیت ۱۲۰ (۱/۸۶ ثانیه) است. این نتایج نشان می‌دهد که استراتژی پیشنهادی بسیار سریع است و افزایش جمعیت تأثیر بسیار ناچیزی بر زمان دارد.

۴- ارزیابی

در این قسمت نتایج حاصل از پیاده‌سازی استراتژی TLBO بررسی می‌شود. دو معیار مهمی که در آرایه پوشش بررسی می‌شود، دقت الگوریتم است که هرچقدر اندازه دنباله آزمون تولید شده کمینه باشد دقت الگوریتم بالاتر است. معیار دیگر، سرعت الگوریتم است که هرچقدر الگوریتم سرعت بالایی داشته باشد درنتیجه توانایی تولید آرایه پوشش برای پیکربندی‌های با ابعاد بزرگ‌تری را دارد که شامل مقادیر بالای قوه پوشش t، v و k است که مهم‌ترین آن‌ها t است. توانایی راهکار پیشنهادی با پیاده‌سازی الگوریتم در پیکربندی‌های متعدد بررسی می‌شود. نتایج با کارآمدترین روش‌های فرامکاشفه‌ای و سایر استراتژی‌های موجود مقایسه می‌شود و برتری آن نشان داده می‌شود. بهترین و میانگین نتایج برای هر CA در ۲۰ بار تکرار الگوریتم در جدول‌ها آورده شده است. محیط اجرایی ارزیابی‌های انجام‌شده متشکل از: Windows 7، CPU 2.20GHz core™ i7QM و RAM 6GB و کدنویسی و پیاده‌سازی استراتژی TLBO در محیط MATLAB R2013b انجام شده است. در جدول‌ها NA[۱۵] نشان‌دهنده موجود نبودن نتایج در مقالات معتبر و یا این‌که نتایج بیش از یک روز تولید می‌شوند و NS[۱۶] نشان‌دهنده عدم توانایی استراتژی در تولید پیکربندی مربوطه است. اعداد هاشورخورده در جدول‌ها نشان‌دهنده بهترین نتایج برای پیکربندی مربوطه است.

جدول‌های ۳ تا ۶ نتایج ارزیابی‌های انجام‌شده را نشان می‌دهد. به‌دلیل سرعت جستجوی بسیار بالا، الگوریتم توانایی تولید نتایج تا قوه ۱۵ را دارد. جدول‌های ۳ تا ۵ آرایه پوشش با قوه ثابت را نشان می‌دهند. جدول ۶ مربوط به ساختار با قوه متغیر است. ما ارزیابی‌های موجود در مقالات معتبر [۲۰، ۲۱، ۲۸] را گسترش داده‌ایم و کارمان را با آن‌ها مقایسه کرده‌ایم. جدول ۳ اندازه آرایه پوشش را نشان می‌دهد، برای k پارامتر که هرکدام ۳ مقدار دارند و قوه پوشش (t) از ۲ تا ۶ گسترش داده شده است. استراتژی CTE-XL فقط تا t = ۳ قادر به تولید دنباله آزمون است که ضعیف‌ترین استراتژی از لحاظ قوه می‌باشد، استراتژی ITCH قادر به تولید نتایج تا t = ۴ است و در ۱۰ مورد بهترین دقت را دارد. الگوریتم B-H به‌دلیل طولانی بودن فرآیند جستجو فقط در مواردی که پیکربندی‌ها اندازه کوچکی دارند قادر به تولید نتایج است. در این‌جا تا قوه t = ۴ را پشتیبانی می‌کند و در ۱۰ مورد قادر به تولید کم‌ترین اندازه دنباله آزمون است. با افزایش t نتایج استراتژی‌های مبتنی بر محاسبات، ضعیف‌تر و رقابت بین سه استراتژی مبتنی بر هوش مصنوعی CS، PSTG و TLBO بیش‌تر می‌شود. در حالت کلی IPOG-D، TConfig و Jenny در ۱ مورد، CS در ۱۶ مورد، PSTG در ۱۰ مورد و استراتژی پیشنهادی در ۲۵ مورد بهترین نتایج را تولید می‌کنند.

جدول ۴ نتایج ارزیابی پیکربندی با ۷ پارامتر که مقادیر آن $۲\leq v\leq ۱۰$ و قوه آن $۲\leq t\leq ۴$ است را نشان می‌دهد. دلیل این ارزیابی نشان دادن تأثیر افزایش v بر روی اندازه دنباله آزمون تولیدشده است. افزایش v فضای جستجوی مسئله را بسیار افزایش می‌دهد و عملاً جستجو را برای بسیاری از استراتژی دشوار می‌کند. به همین دلیل نتایج اکثر استراتژی‌ها تا v = ۵ در دسترس است، لذا ابتدا به بررسی $۲\leq v\leq ۵$ پرداخته می‌شود. استراتژی‌های Jenny، PICT و TVG در این پیکربندی بسیار ضعیف هستند و در هیچ حالتی دنباله آزمون بهینه تولید نمی‌کنند. استراتژهای IPOG-D و TConfig نیز ضعیف عمل می‌کنند و فقط در ۱ مورد نتیجه مناسبی دارند. استراتژی ITCH در ۴ مورد نمونه آزمون کمینه را تولید می‌کند و در سایر موارد نتایج قابل قبولی را دارد. در این ارزیابی نیز استراتژی‌های مبتنی بر هوش مصنوعی قوی‌تر هستند. استراتژی PSTG مقداری ضعیف‌تر از پیکربندی قبل عمل می‌کند و فقط در ۲ مورد دنباله آزمون کمینه را تولید می‌کند. استراتژی CS در۶ مورد، H-B در ۵ مورد و TLBO در ۸ مورد بهتر از سایر استراتژی‌ها است. برای $۵\leq v\leq ۱۰$ نتایج سایر استراتژی‌ها در دسترس نیست و تنها نتایج استراتژی پیشنهادی قابل مشاهده است.

در جدول ۵ تأثیر افزایش تعداد مقادیر در بازه $۲\leq v\leq ۱۰$ بر اندازه دنباله آزمون تولیدشده برای ساختار CA(N; ۴,۱۰, v) بررسی می‌شود. در v = ۲ استراتژی CS، در v = ۴ و v = ۱۰ استراتژی GTWay نتایج کمینه را تولید می‌کنند و TLBO از ۸ پیکربندی متفاوت در ۵ مورد بهترین نتایج را تولید می‌کند.

جدول ۶ مربوط به ساختار VSCA است. الگوریتم PSTG نتایج بسیار خوبی را تولید می‌کند اما تا t = ۶ قادر به تولید دنباله آزمون است. استراتژی‌های مبتنی بر محاسبات عملکرد ضعیفی دارند و استراتژی‌های TVG و WITCH یک مورد کمینه را تولید می‌کنند و IPOG با تولید سه مقدار کمینه عملکرد بهتری دارد. الگوریتم TLBO تقریباً در تمامی موارد جواب کمینه را تولید می‌کند و توانایی پشتیبانی تا t = ۱۵ را دارد. کم‌تر الگوریتمی قادر به تولید برای هر دو ساختار آرایه پوشش با قوه ثابت و قوه متغیر است. استراتژی TLBOبا دقت بسیار بالا در هر دو ساختار هم از نظر تولید دنباله آزمون کمینه و هم از لحاظ پشتیبانی از تولید پیکربندی‌هایی با ابعاد بزرگ و قوه t = ۱۵ توانمندی خود را نشان می‌دهد.

۵- نتیجه‌گیری

تولید آرایه پوشش یک مسئله کلیدی در حوزه آزمون‌های ترکیباتی است. در این مقاله از الگوریتم فرامکاشفه‌ای مبتنی بر آموزش و یادگیری (TLBO) برای ساخت آرایه پوشش استفاده می‌شود. در این کار با طراحی ابتکاری تابع برازندگی فضای جستجو را کاهش داده‌ایم. به همین دلیل سرعت استراتژی TLBO بسیار بالا رفته است، به‌گونه‌ای که توانایی پشتیبانی تا قوه t = ۱۵ را دارد. علاوه‌بر سرعت بالا، دقت الگوریتم نیز بالا است و قادر به تولید دنباله آزمون بهینه در اکثر موارد است. راهکار پیشنهادی را با سایر استراتژی‌های فرامکاشفه‌ای و الگوریتم‌های مبتنی بر محاسبات مقایسه کرده‌ایم و نشان داده‌ایم که الگوریتم TLBO از لحاظ اندازه آرایه در بسیاری از پیکربندی‌ها بهترین نتایج را تولید می‌کند. همچنین این استراتژی بسیار سریع‌تر از سایر استراتژی‌های مبتنی بر هوش مصنوعی می‌باشد و توان تولید نتایج را در پیکربندی‌های

بزرگ تا t = ۱۵ را دارد. در کارهای آینده می‌توان این پژوهش را به روش‌های متعددی بهبود داد. روش اول، ترکیب الگوریتم TLBO با سایر الگوریتم‌های فرامکاشفه‌ای است. دومین روش، به‌جای استفاده از رویکرد یک-آزمون-در-یک-زمان از رویکرد یک-پارامتر-در-یک-زمان استفاده کنیم. راهکار سوم، ترکیب الگوریتم TLBO با روش‌های مبتنی بر محاسبات است.

جدول ۳: اندازه دنباله آزمون تولیدی برای CA(N; t, k, ۳) ۱۲≤ k ≤۳ و ۶≤ t ≤ ۲

t	k	Jenny N	TConfig N	ITCH N	PICT N	TVG N	IPOG-D N	IPOG N	PSTG N	CS N	H-B N	**TLBOS** N.Best	**TLBOS** N.AVG
۲	۳	۹	۱۰	۹	۱۰	۱۰	۱۵	۹	۹	۹	۹	۹	۹/۹
	۴	۱۳	۱۰	۹	۱۳	۱۲	۱۵	۹	۹	۹	۹	۹	۱۰/۰
	۵	۱۴	۱۴	۱۵	۱۳	۱۳	۱۵	۱۵	۱۲	۱۱	۱۲	۱۱	۱۲/۵
	۶	۱۵	۱۵	۱۵	۱۴	۱۵	۱۵	۱۵	۱۳	۱۳	۱۳	۱۳	۱۴/۳
	۷	۱۶	۱۵	۱۵	۱۶	۱۵	۱۵	۱۵	۱۵	۱۴	۱۴	۱۴	۱۵/۱
	۸	۱۷	۱۷	۱۵	۱۶	۱۵	۱۵	۱۵	۱۵	۱۵	۱۵	۱۵	۱۵/۹
	۹	۱۸	۱۷	۱۵	۱۷	۱۵	۱۵	۱۵	۱۷	۱۶	۱۶	۱۵	۱۶/۴
	۱۰	۱۹	۱۷	۱۵	۱۸	۱۶	۲۱	۱۵	۱۷	۱۷	۱۷	۱۶	۱۶/۹
	۱۱	۱۷	۲۰	۱۵	۱۸	۱۶	۲۱	۱۷	۱۷	۱۸	۱۷	۱۶	۱۷/۶
	۱۲	۱۹	۲۰	۱۵	۱۹	۱۶	۲۱	۲۱	۱۸	۱۸	۱۷	۱۷	۱۸/۱
۲	۴	۳۴	۳۲	۲۷	۳۴	۳۴	۲۷	۳۲	۲۷	۲۸	۲۸	۲۷	۳۰/۶
	۵	۴۰	۴۰	۴۵	۴۳	۴۱	۴۵	۴۱	۳۹	۳۸	۳۸	۳۸	۳۹/۶
	۶	۵۱	۴۸	۴۵	۴۸	۴۹	۴۵	۴۶	۴۵	۴۳	۴۳	۴۳	۴۵/۷
	۷	۵۱	۵۵	۴۵	۵۱	۵۵	۵۰	۵۵	۵۰	۴۸	۴۹	۴۹	۵۱/۱
	۸	۵۸	۵۸	۴۵	۵۹	۶۰	۵۰	۵۶	۵۴	۵۳	۵۳	۵۴	۵۵/۱
	۹	۶۲	۶۴	۷۵	۶۳	۶۴	۷۱	۶۳	۵۸	۵۸	۵۷	۵۸	۵۹/۰
	۱۰	۶۵	۶۸	۷۵	۶۵	۶۸	۷۱	۶۶	۶۲	۶۲	۶۲	۶۱	۶۲/۴
	۱۱	۶۵	۷۲	۷۵	۷۰	۶۹	۷۶	۷۰	۶۴	۶۶	۶۴	۶۳	۶۵/۷
	۱۲	۶۸	۷۷	۷۵	۷۲	۷۰	۷۶	۷۳	۶۷	۷۰	۶۶	۶۶	۶۷/۷
۴	۵	۱۰۹	۹۷	۱۵۳	۱۰۰	۱۰۵	۱۶۲	۹۷	۹۶	۹۴	۹۳	۹۲	۹۶/۲
	۶	۱۴۰	۱۴۱	۱۵۳	۱۴۲	۱۳۹	۱۶۲	۱۴۱	۱۳۳	۱۳۲	۱۳۲	۱۳۱	۱۳۳/۲
	۷	۱۶۹	۱۶۶	۲۱۶	۱۶۸	۱۷۲	۲۲۶	۱۶۷	۱۵۵	۱۵۴	۱۵۴	۱۵۴	۱۵۸/۳
	۸	۱۸۷	۱۹۰	۲۱۶	۱۸۹	۱۹۲	۲۲۶	۱۹۲	۱۷۵	۱۷۳	۱۷۵	۱۷۵	۱۷۷/۱
	۹	۲۰۶	۲۱۳	۳۰۶	۲۱۱	۲۱۵	۲۶۰	۲۱۰	۱۹۵	۱۹۵	۱۹۵	۱۹۳	۱۹۵/۳
	۱۰	۲۲۱	۲۳۵	۳۳۶	۲۳۱	۲۳۳	۲۷۸	۲۳۳	۲۱۰	۲۱۱	۲۱۰	۲۰۹	۲۱۱/۷
	۱۱	۲۳۶	۲۵۸	۳۴۸	۲۴۷	۲۵۰	۳۳۲	۲۵۱	۲۲۲	۲۲۹	۲۲۵	۲۲۶	۲۲۸/۴
	۱۲	۲۵۲	۲۷۲	۳۷۲	۱۹۶	۲۶۸	۳۳۲	۲۷۲	۲۴۴	۲۵۳	۲۴۵	۲۴۱	۲۴۴/۶
۵	۶	۳۴۸	۳۰۵	NA	۳۱۰	۳۲۱	۳۸۶	۳۰۵	۳۱۲	۳۰۴	NA	۳۰۴	۳۱۲/۲
	۷	۴۵۸	۴۷۷	NA	۴۵۲	۴۶۲	۶۷۸	۴۶۶	۴۴۱	۳۴۳	NA	۴۴۱	۴۴۴/۶
	۸	۵۴۸	۵۸۳	NA	۵۵۵	۵۶۲	۷۵۶	۵۷۵	۵۱۵	۵۱۵	NA	۵۱۵	۵۱۹/۳
	۹	۶۳۳	۶۸۴	NA	۶۳۷	۶۶۰	۱۰۴۳	۶۶۷	۵۹۸	۵۹۰	NA	۵۹۴	۶۰۰/۱
	۱۰	۷۱۴	۷۷۳	NA	۷۳۵	۷۵۰	۱۱۱۸	۷۶۱	۶۶۷	۶۸۲	NA	۶۷۲	۶۷۷/۵
	۱۱	۷۹۱	۸۵۸	NA	۸۲۲	۸۳۳	۱۳۷۲	۸۵۱	۷۴۷	۷۷۸	NA	۷۴۱	۷۴۶/۳
	۱۲	۸۵۰	۹۳۸	NA	۹۰۰	۸۲۴	۱۴۴۹	۹۲۹	۸۰۹	۸۸۰	NA	۸۰۶	۸۱۱/۷
۶	۷	۱۰۸۶	۹۲۱	NA	۱۰۱۵	۱۰۲۷	۱۲۰۱	۹۲۱	۹۷۷	۹۶۳	NA	۹۶۸	۹۸۲/۶
	۸	۱۴۶۶	۱۵۱۵	NA	۱۴۵۵	۱۴۴۸	۱۷۶۳	۱۴۹۳	۱۴۰۲	۱۴۰۱	NA	۱۳۹۷	۱۴۰۳/۲
	۹	۱۸۴۰	۱۹۳۱	NA	۱۸۱۸	۱۸۴۹	۲۵۲۶	۱۸۸۹	۱۶۸۴	۱۶۸۹	NA	۱۶۹۲	۱۷۰۵/۲
	۱۰	۲۱۶۰	NA	NA	۲۱۶۵	۲۱۹۲	۲۸۳۴	۲۲۶۲	۱۹۸۰	۲۰۲۷	NA	۱۹۸۰	۱۹۹۰/۹
	۱۱	۲۴۵۹	NA	NA	۲۴۹۶	۲۵۳۳	۳۸۸۶	۲۶۰۷	۲۲۵۵	۲۲۹۸	NA	۲۲۶۹	۲۲۷۳/۶
	۱۲	۲۷۵۷	NA	NA	۲۸۱۵	۲۵۹۷	۴۰۸۷	۳۶۴۹	۲۵۲۸	۲۶۳۸	NA	۲۵۴۱	۲۵۴۸/۶

جدول ۴: اندازه دنباله آزمون تولیدی برای CA(N; t, ۷, v) ۲≤ v ≤۱۰ و ۲ ≤ t ≤۴

t	v	Jenny N	TConfig N	ITCH N	PICT N	TVG N	IPOG-D N	IPOG N	PSTG N	CS N	H-B N	TLBOS N.Best	TLBOS N.AVG
	۲	۸	۷	۶	۷	۷	۸	۷	۶	۶	۶	۶	۷/۳
	۳	۱۶	۱۵	۱۵	۱۶	۱۵	۱۵	۱۵	۱۵	۱۵	۱۵	۱۵	۱۵/۴
	۴	۲۸	۲۸	۲۸	۲۷	۲۷	۳۵	۲۹	۳۶	۲۵	۲۵	۲۴	۲۶/۱
	۵	۳۷	۴۰	۴۵	۴۰	۴۲	۴۵	۴۵	۳۷	۳۷	۳۶	۳۷	۳۸/۱
۲	۶	۵۵	۵۷	NA	۵۶	NA	۷۲	۵۵	NA	NA	NA	۵۳	۵۴/۲
	۷	۷۴	۷۶	NA	۷۴	NA	۹۱	۴۹	NA	NA	NA	۷۱	۷۲/۷
	۸	۹۳	۱۰۱	NA	۹۷	NA	۱۲۸	۹۵	NA	NA	NA	۹۲	۹۵
	۹	۱۱۴	۱۲۴	NA	۱۲۱	NA	۱۵۹	۱۰۷	NA	NA	NA	۱۱۷	۱۱۸/۸
	۱۰	۱۵۷	۱۵۲	NA	۱۴۸	NA	۲۰۰	۱۳۹	NA	NA	NA	۱۳۹	۱۴۲/۸
	۲	۱۴	۱۶	۱۳	۱۵	۱۵	۱۴	۱۶	۱۳	۱۲	۱۳	۱۲	۱۴/۲
	۳	۵۱	۵۵	۴۵	۵۱	۵۵	۵۰	۵۵	۵۰	۴۹	۴۹	۴۹	۵۵/۱
	۴	۱۲۴	۱۱۲	۱۱۲	۱۲۴	۱۳۴	۱۱۴	۱۱۲	۱۱۶	۱۱۷	۱۱۶	۱۱۶	۱۹/۶
	۵	۲۳۶	۲۳۶	۲۲۵	۲۴۱	۲۶۰	۲۵۲	۲۳۷	۲۲۵	۲۲۳	۲۲۴	۲۲۴	۲۲۶/۲
۳	۶	۴۰۰	۴۲۳	NA	۴۱۳	NA	۴۷۰	۴۲۰	NA	NA	NA	۳۸۳	۳۸۵/۶
	۷	۶۲۴	۶۵۲	NA	۶۴۳	NA	۶۹۲	۶۴۹	NA	NA	NA	۶۰۴	۶۰۷/۳
	۸	۹۲۴	۹۶۴	NA	۹۵۷	NA	۱۰۱۶	۹۳۵	NA	NA	NA	۸۹۷	۹۰۰/۴
	۹	۱۲۸۸	۱۳۷۷	NA	۱۲۴۱	NA	۱۴۹۶	۱۳۶۶	NA	NA	NA	۱۲۷۶	۱۲۸۰/۲
	۱۰	۲۱۷۲	۱۸۸۸	NA	۱۸۱۵	NA	۲۲۱۸	۱۸۶۲	NA	NA	NA	۱۷۵۱	۱۷۵۵/۱
	۲	۳۱	۳۶	۴۰	۳۲	۳۱	۴۰	۳۵	۲۹	۲۷	۲۷	۲۷	۳۰/۲
	۳	۱۶۹	۱۶۶	۲۱۶	۱۶۸	۱۶۸	۲۲۶	۱۶۷	۱۵۵	۱۵۴	۱۵۴	۱۵۴	۱۵۸/۳
	۴	۵۱۷	۵۶۸	۷۰۴	۵۲۹	۵۵۹	۷۰۴	۶۱۴	۴۸۷	۴۸۷	۴۹۰	۴۸۶	۴۸۹/۶
	۵	۱۲۴۸	۱۳۲۰	۱۷۵۰	۱۲۷۹	۱۳۸۵	۱۸۵۸	۱۲۹۳	۱۱۷۶	۱۱۷۱	۱۱۸۲	۱۱۶۹	۱۱۷۷/۲
۴	۶	۲۵۶۰	۵۰۷۵	NA	NA	NA	۳۸۹۲	۲۶۹۷	NA	NA	NA	۲۴۰۷	۲۴۱۳/۵
	۷	۴۶۸۴	NA	NA	NA	NA	۷۱۹۰	۴۹۷۷	NA	NA	NA	۴۴۳۰	۴۴۴۲/۳
	۸	۷۸۹۸	NA	NA	NA	NA	۱۱۷۷۶	۸۲۳۷	NA	NA	NA	۷۵۳۲	۴۵۵۳/۸
	۹	۱۲۵۳۴	NA	NA	NA	NA	۱۹۹۰۶	۱۳۵۱۳	NA	NA	NA	۱۱۹۴۱	۱۱۹۶۴/۲
	۱۰	۲۶۶۲۴	NA	NA	NA	NA	۳۱۰۵۲	۲۰۵۸۴	NA	NA	NA	۱۸۱۳۶	۱۸۱۸۰/۲

جدول۵: اندازه دنباله آزمون تولیدی برای CA(N; ۴,۱۰, v) ۲≤ v ≤۱۰ و

v	Jenny N	TConfig N	ITCH N	PICT N	TVG N	IPOG N	GTWay N	PSTG N	CS N	TLBOS N.Best	TLBOS N.AVG
۲	۳۹	۴۵	۵۸	۴۳	۴۰	۴۵	۴۶	۳۴	۲۸	۳۷	۳۸/۹
۳	۲۲۱	۲۳۵	۳۳۶	۲۳۱	۲۲۸	۲۳۳	۲۲۴	۲۱۳	۲۱۱	۲۱۱	۲۱۲/۸
۴	۷۰۳	۷۱۸	۷۰۴	۷۲۴	۷۸۲	۷۱۵	۶۲۱	۶۸۵	۶۹۸	۶۷۶	۶۷۹/۲
۵	۱۷۷۹	۱۸۷۵	۱۷۵۰	۱۸۱۲	۱۹۱۷	۱۸۶۹	۱۷۱۴	۱۷۱۶	۱۷۳۱	۱۶۶۲	۱۶۷۰/۵
۶	۳۵۱۹	NA	NA	۳۷۳۵	۴۱۹۵	۳۸۹۱	۳۵۱۴	۳۸۸۰	۳۸۹۴	۳۴۴۲	۳۴۴۸/۲
۷	۶۴۸۲	NA	NA	۶۸۸۶	NA	۷۱۵۳	۶۴۵۹	NA	NA	۶۳۶۶	۶۳۹۰/۷
۸	۱۱۰۲۱	NA	NA	۱۱۶۳۰	NA	۱۲۱۰۳	۱۰۸۵۰	NA	NA	۱۰۸۲۴	۱۰۸۳۶/۲
۹	۱۷۵۲۷	NA	NA	۱۸۵۰۷	NA	۱۹۴۱۹	۱۷۲۷۲	NA	NA	۱۷۲۴۹	۱۷۲۷۰/۸
۱۰	۲۶۶۲۴	NA	NA	۲۸۰۲۵	NA	۲۹۵۴۱	۲۶۱۲۱	NA	NA	۲۶۳۶۴	۲۶۳۸۹/۴

جدول ۶: اندازه دنباله آزمون تولیدی برای(VSCA(N; ۳,۱۵, ۳, {C)

C	PICT N	TVG N	IPOG N	WHITCH N	SA N	ACS N	VS-PSTG N	**TLBOS** N.Best	**TLBOS** N.AVG
Ø	۸۳	۸۴	۸۲	۷۵	NS	NS	۷۵	۷۵	۷۷/۳
CA(۴, ۴, ۳)	۱۵۰۷	۹۳	۸۷	۱۲۹	NS	NS	۹۱	۸۸	۸/۸
CA(۵, ۵, ۳)	۵۳۶۶	۲۴۴	۲۴۳	۲۷۳	NS	NS	۲۴۳	۲۴۳	۲۴۸/۶
CA(۶, ۶, ۳)	۱۲۶۰۹	۷۲۹	۷۲۹	۷۵۹	NS	NS	۷۲۹	۷۲۹	۷۲۹
CA(۷, ۷, ۳)	NA	NA	۲۱۸۷	NA	NS	NS	NS	۲۱۸۷	۲۱۸۷
CA(۴, ۵, ۳)	۱۷۹۳	۱۱۸	۱۱۹	۱۵۱	NS	NS	۱۱۴	۱۱۴	۱۱۸/۴
CA(۵, ۶, ۳)	۵۳۸۷	۳۲۳	۳۳۷	۲۸۷	NS	NS	۳۱۴	۳۱۲	۳۱۸/۹
CA(۶, ۷, ۳)	۱۶۷۹۲	۱۰۱۸	۱۲۱۵	۱۴۴	NS	NS	۱۰۰۲	۹۸۵	۹۹۰/۴
CA(۴, ۷, ۳)	۲۷۸۱	۱۶۸	۱۸۳	۲۱۹	NS	NS	۱۵۹	۱۵۸	۱۶۱/۴
CA(۴, ۹, ۳)	۳۰۵۹	۲۱۴	۲۲۷	۲۸۹	NS	NS	۱۹۵	۱۹۵	۲۰۲/۵
CA(۴, ۱۱, ۳)	۲۸۲۴	۲۵۶	۲۵۹	۳۵۴	NS	NS	۲۲۶	۲۲۶	۲۳۲/۱
CA(۴, ۱۵, ۳)	NS	۳۲۷	۴۹۸	۴۹۸	NS	NS	۲۸۴	۲۸۴	۲۹۰/۸
CA(۵, ۷, ۳)	۷۴۷۵	۴۷۱	۷۱۳	۴۸۱	NS	NS	۴۳۷	۴۳۷	۴۴۴/۲
CA(۵, ۱۱, ۳)	۸۶۹۰	۵۵۶	۷۱۴	۶۲۰	NS	NS	۵۱۶	۵۱۶	۵۲۳/۲
CA(۶, ۱۱, ۳)	۲۲۸۳۳	۱۴۷۹	۲۱۰۸	۱۵۱۳	NS	NS	۱۳۹۶	۱۳۹۸	۱۴۰۵/۹
CA(۶, ۹, ۳)	۲۶۷۲۹	۱۸۴۰	۲۱۲۴	۱۹۶۴	NS	NS	۱۶۹۰	۱۶۸۹	۱۶۹۷/۷
CA(۸, ۸, ۳)	NA	NA	۶۵۶۱	NA	NS	NS	NS	۶۵۶۱	۶۵۶۱
CA(۹, ۹, ۳)	NA	NA	۱۹۶۸۳	NA	NS	NS	NS	۱۹۶۸۳	۱۹۶۸۳
CA(۱۰, ۱۰, ۳)	NA	NA	۵۹۰۴۹	NA	NS	NS	NS	۵۹۰۴۹	۵۹۰۴۹
CA(۱۱, ۱۱, ۳)	NA	NA	۱۷۷۱۴۷	NA	NS	NS	NS	۱۷۷۱۴۷	۱۷۷۱۴۷
CA(۱۲, ۱۲, ۳)	NA	NA	۵۳۱۴۴۱	NA	NS	NS	NS	۵۳۱۴۴۱	۵۳۱۴۴۱
CA(۱۳, ۱۳, ۳)	NA	NA	۱۵۹۴۳۲۳	NA	NS	NS	NS	۱۵۹۴۳۲۳	۱۵۹۴۳۲۳
CA(۱۴, ۱۴, ۳)	NA	NA	۴۷۸۲۹۶۹	NA	NS	NS	NS	۴۷۸۲۹۶۹	۴۷۸۲۹۶۹

مراجع

[1] C. Nie and H. Leung, "A survey of combinatorial testing," *ACM Computing Surveys.*, vol. 43, no. 2, pp.1–11, 2011.

[2] D. Kuhn and M. Reilly, "An investigation of the applicability of design of experiments to software testing," *Software Engineering Workshop*, 2002. Proceedings. 27th Annual NASA Goddard/IEEE, pp. 91–95, NASA Goddard Space Flight Center, 2002.

[3] W. Afzal, R. Torkar, R. Feldt, "A systematic review of search-based testing for non-functional system properties," *Information and Software Technology*, vol. 51, no. 6, pp.957–976, 2009.

[4] C. Yilmaz, M.B. Cohen and A. Porter, "Covering arrays for efficient fault characterization in complex configuration spaces," *ACM SIGSOFT Software Engineering Notes*, vol.29, no.4, 2004.

[5] A. Ouaarab, Belaid Ahiod and X.-S. Yang, "Discrete cuckoo search algorithm for the travelling salesman problem," *Neural Computing and Applications*, vol.24, pp.1659-1669, 2014.

[6] M. Chateauneuf and D.L. Kreher, "On the state of strength-three covering arrays,"*Journal of Combinatorial Designs*, vol.10, no. 4 pp.217–238, 2002.

[7] M. Grindal, J. Offutt and S.F. Andler, "Combination testing strategies: a survey," *Software Testing, Verification & Reliability*, vol. 15, no. 3, pp.167–199, 2005.

[8] 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, no. 3, pp.125–148, 2008.

[9] A.W. Williams, "Determination of test configurations for pair-wise interaction coverage," *13th International Conference on Testing Communicating Systems: Tools and Techniques*, The Netherlands, pp. 59–74, 2000.

[10] 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, no. 7, pp.437–444, 1997.

[11] M.B. Cohen, *Designing Test Suites for Software Interaction Testing*, Ph.D. Thesis, University of Auckland, 2004.

[12] J. Czerwonka, "Pairwise testing in real world: practical extensions to test case generator," *Proceedings of 24th Pacific Northwest Software Quality Conference*, pp. 419–430, Citeseer, 2006.

[13] R.C. Bryce and C.J. Colbourn, "The density algorithm for pairwise interaction testing: research articles," *Software Testing, Verification & Reliability*, vol. 17, no. 3, pp.159–182, 2007.

[14] R.C. Bryce and C.J. Colbourn, "A density-based greedy algorithm for higher strength covering arrays," *Software Testing, Verification & Reliability*, vol. 19, no.1, pp.37–53, 2009.

[15] E. Lehmann and J. Wegener, "Test case design by means of the CTE XL," *8th European International Conference on Software Testing, Analysis & Review (EuroSTAR 2000)*, , pp. 1–10, Denmark, 2000.

[16] Y.T. Yu, S.P. Ng and E.Y.K. "Chan, Generating, selecting and prioritizing test cases from specifications with tool support," *3rd International Conference on Quality Software*, pp. 83–90, Dallas, 2003.

[17] Y.-W. Tung and W.S. Aldiwan, "Automating test case generation for the new generation of mission software system," *IEEE Aerospace Conference*, vol.1, pp. 431–437, 2000.

[18] J. Arshem, TVG download page, 2009, http://sourceforge.net/projects/tvg

[19] B. Jenkins, Jenny download web page, 2005, http://burtleburtle.net/bob/math/jenny.html.

[20] M. F. J. Klaib, *Development of An Automated Test Data Generation and Execution Strategy Using Combinatorial Approach*, Ph.D. Thesis, Universiti Sains Malaysia, 2009.

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

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

[23] A. Calvagna and 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*, pp.10–18, Denver, 2009.

[24] M.B. Cohen, M.B. Dwyer and J. Shi, "Interaction testing of highly-configurable systems in the presence of constraints," *International Symposium on Software Testing and Analysis*, pp.129–139, London, 2007.

[25] K.J. Nurmela, "Upper bounds for covering arrays by tabu search," *Discrete applied mathematics*, vol. 138, no. 1, pp.143–152, 2004.

[26] T. Shiba, T. Tsuchiya and T. Kikuno, "Using artificial life techniques to generate test cases for combinatorial testing", *28th Annual International Computer Software and Applications Conference*, vol. 1, pp.72–77, Hong Kong, 2004.

[27] X. Chen, Q. Gu, A. Li and D. Chen, "Variable strength interaction testing with an ant colony system approach," *16th Asia–Pacific Software Engineering Conference*, pp.160–167, Malaysia, 2009.

[28] B. S. Ahmed, K. Z. Zamli, and C. P. Lim, "Application of Particle Swarm Optimization to uniform and variable strength covering array construction," *Applied Soft Computing*, vol.12, no.4, pp.1330–1347, 2012.

[29] B.S. Ahmed, M.A. Sahib and M.Y. Potrus, "Generating combinatorial test cases using Simplified Swarm Optimization (SSO) algorithm for automated GUI functional testing," *Engineering Science and Technology, an International Journal*, vol. 17, no. 4, pp.218–226, 2014.

[30] H. Wu, C. Nie, F. Kuo, H. Leung, and C. J. Colbourn, "A Discrete Particle Swarm Optimization for Covering Array Generation," *IEEE Transactions on Evolutionary Computation*, vol.19, no.4, pp.575–591, 2015.

[31] 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.

[32] A. R. A. Alsewari and 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, no.6, pp.553–568, 2012.

[۳۳] 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، صفحات ۲۵-۳۵، ۱۳۹۵.

[34] M. H. M. Zabil and K. Z. Zamli, "Implementing a T-Way Test Generation Strategy Using Bees Algorithm," *International Journal of Soft Computing and Its Applications*, vol.5, no.3, pp.116-126, 2013.

[35] Y. A. Alsariera, M. A. Majid, and K. Z. Zamli, "A Bat-inspired strategy for pairwise testing," *Advanced Science Letters*, vol.10, no.18, pp.8500–8506, 2015.

[36] V. Bhargava, S.E.K. Fateen and A. Bonilla-Petriciolet, "Cuckoo Search: a new natureinspired optimization method for phase equilibrium calculations," *Fluid Phase Equilib.*, vol.337, pp.191–200, 2013.

[37] Zamli, Kamal Z., Basem Y. Alkazemi, and Graham Kendall. "A Tabu Search hyper-heuristic strategy for t-way test suite generation," *Applied Soft Computing*, vol.44, pp.57-74, 2016.

[38] B. Stevens and E. Mendelsohn, "Efficient software testing protocols," *Proceedings ofthe 8th IBM Centre for Advanced Studies Conference (CASCON'98)*, pp.279–293, Toronto, 1998.

[39] 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*, vol.43, no.3, pp.303–315, 2011.

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

زیرنویس‌ها

۱ Test case generatin
۲ Combinatorial testing
۳ Covering array
۴ T-way
۵ NP hard optimization problem
6 Mathmathical methode
۷ Computational methode
۸ Orthogonal arrays
۹ One-test-at-a-time
۱۰ One-parameter-at-a-time
۱۱ Search Based Software Eengineering
۱۲ Teaching learning base optimization
۱۳ Exhaustive testing
۱۴ Variable strength covering array
۱۵ Not available
۱۶ Not supported