مجله محاسبات نرم
SOFT COMPUTING JOURNAL
تارنمای مجله: scj.kashanu.ac.ir
دانشگاه کاشان
University of Kashan

استفاده از الگوریتم تودهٔ ذرات بهینه در تولید دنبالهٔ آزمون کمینه در آرایهٔ پوشش با قوهٔ ثابت✧

سجاد اسفندیاری[۱*]، دانشجوی دکتری، وحید رافع[۲]، دانشیار
[۱ و ۲] دانشکده فنی مهندسی، گروه مهندسی کامپیوتر دانشگاه اراک، اراک، ایران.

اطلاعات مقاله

تاریخچه مقاله:
دریافت ۰۹ اسفند ماه ۱۳۹۸
پذیرش ۱۰ مهر ماه ۱۳۹۹

کلمات کلیدی:
آزمون ترکیبی
تولید داده آزمون
آرایهٔ پوشش
الگوریتم تودهٔ ذرات بهینه

چکیده

تاکنون، تعداد زیادی الگوریتم مفید برای تولید آرایه پوشش که یکی از شاخه‌های آزمون ترکیبی است، ارایه شده است. اصلی‌ترین چالش در تولید چنین آرایه‌هایی، تولید آرایه با تعداد نمونه آزمون کمینه (بهره‌وری) در زمان مناسب (کارایی)، برای سیستم‌های بزرگ است. استراتژی‌های تولید آرایه پوشش قالبا به دو دسته کلی محاسباتی و فرامکاشفه‌ای تقسیم می‌شوند. استراتژی‌های محاسباتی معمولا کارایی بالایی دارند اما از نظر بهره‌وری نتایج ضعیفی را تولید می‌کند و استراتژی‌های فرامکاشفه‌ای از بهره‌وری مناسب و کارایی ضعیفی برخوردار هستند. در بین استراتژی‌های در دسترس استراتژی DPSO از نظر بهره‌وری بهترین نتایج را دارد، اما کارایی مناسبی ندارد و استراتژی GS کارایی مناسبی را دارد اما بهره‌وری مناسبی ندارد. در حالت کلی استراتژی که کارایی و بهره‌وری را توامان داشته باشد، به چشم نمی‌خورد. در این مقاله ما سعی کردیم با استفاده از توده ذرات دنباله آزمون مناسب را از نظر بهره‌وری و کارایی، تولید کنیم. همچنین برای افزایش بهره‌وری از یک تابع کمینه‌ساز ساده و تاثیرگذار استفاده شده است. نتایج ارزیابی نشان می‌دهد که راهکار پیشنهادی از نظر کارایی و بهره‌وری نتایج مطلوب دارد.

۱. مقدمه

سیستم‌های نرم‌افزاری پارامترهای مختلفی دارند. هر پارامتر تعدادی مقدار را به خود اختصاص می‌دهد. در آزمون کامل نیاز است که تمام حالت‌های ممکن آزمون شود. برای مثال اگر سیستمی ۹ پارامتر داشته باشد و هریک از پارمترها ۵ مقدار را داشته باشند، در حالت کلی این سیستم ۵۹ (۱، ۹۵۳، ۱۲۵) ورودی مختلف دارد که باید آزمون شوند. با افزایش تعداد این مقادیر عملاً آزمون کامل سیستم غیرممکن می‌شود [۱] و می‌توان گفت که انفجار محاسباتی در آزمون رخ می‌دهد. آزمون ترکیبی راهکاری مناسب برای جلوگیری از این انفجار است [۲]. آرایهٔ پوشش[1] یکی از رایج‌ترین آزمون‌های آزمون ترکیبی است که به آزمون t-way نیز معروف است که در آن t قوه تعامل است [۳]. در آزمون t-way به‌جای بررسی کامل تعاملات یک سیستم تمام ترکیب‌های t تایی آن سیستم مورد نیاز است.

آرایهٔ پوشش با بهره‌گیری از راهکارهای محاسباتی و هوش مصنوعی، در واقع با نمونه‌برداری تعدادی از نمونهٔ آزمون‌ها سعی در پوشش کامل این ترکیبات دارد. هرچه تعداد این

✧ نوع مقاله: پژوهشی
* نویسنده مسئول
پست‌های الکترونیک: sajad.a1367@gmail.com (اسفندیاری)
v-rafe@araku.ac.ir (رافع)

1. Covering Array (CA)

نحوه ارجاع به مقاله: اسفندیاری، سجاد، رافع، وحید، «استفاده از الگوریتم تودهٔ ذرات بهینه در تولید دنبالهٔ آزمون کمینه در آرایهٔ پوشش با قوهٔ ثابت»، مجله محاسبات نرم، جلد ۸، شماره ۲، ص۶۶ـ۷۹، پاییز و زمستان ۱۳۹۸.

نمونه‌ها (نمونهٔ آزمون) کمتر باشد، راهکار انتخاب نمونهٔ آزمون قوی‌تر است [۲]؛ دلیل آن نیز این است که با تعداد کمتر نمونهٔ آزمون در آینده می‌توان سیستم را آزمایش کرد. در حالت کلی برای آرایهٔ پوشش سه چالش وجود دارد: ۱. انتخاب نمونهٔ آزمون کمینه (بهره‌وری) (۲) زمان اجرای راهکار (کارایی) و (۳) توانایی در پوشش قوهٔ تعامل بالا [۲]. عمده راهکارهای تولید آرایهٔ پوشش به دو دستهٔ کلی مبتنی بر محاسبات محض و هوش مصنوعی تقسیم می‌شوند. تعداد زیادی استراتژی‌های مبتنی بر محاسبات وجود دارند که از نظر کارایی بسیار مناسب‌اند، اما بهره‌وری‌شان در سطح استراتژی‌های هوش مصنوعی نیست. بهترین استراتژی مبتنی بر محاسبات از نظر کارایی IPOG[1] [۴] است و استراتژی CTX[2] [۵] نیز از نظر بهره‌وری بهترین عملکرد را در بین استراتژی‌های محاسباتی دارد اما کارایی مناسبی ندارد و تا $t = 4$ توان تولید آرایهٔ پوشش را دارد. رقابت اصلی از نظر بهره‌وری آرایهٔ پوشش بین استراتژی‌های هوش مصنوعی است که تعداد آن‌ها بسیار زیاد است. اصلی‌ترین مشکل در تولید دنبالهٔ آزمون کمینه برای آرایهٔ پوشش با استفاده از الگوریتم‌های فرامکاشفه‌ای، گیر کردن در بهینهٔ محلی است. برای رفع این مشکل استراتژی DPSO[3] راهکار مناسبی را ارائه داد و بهترین نتایج را از نظر بهره‌وری ارائه داد اما از نظرکارایی نتایج مناسبی را نداشت [۲]. همچنین تا $t = 10$ توان تولید دنبالهٔ آزمون را دارد. البته با توجه به اینکه پشتیبانی از قوهٔ تعامل بالا رابطهٔ مستقیمی با کارایی دارد ممکن است از DPSO سخت‌افزارهای پرقدرت $t > 10$ را نیز پشتیبانی کند. استراتژی‌های مختلفی با بهره‌وری مناسب ارائه شده که نتایج برابری را با DPSO داشتند و با توجه به در دسترس نبودن این ابزارها نمی‌توان در خصوص کارایی‌شان اظهار نظر کرد. استراتژی GS[4] نیز از دیگر استراتژی مبتنی بر هوش مصنوعی است که توان مناسبی از نظر کارایی دارد و تا $t = 20$ را پشتیبانی می‌کند اما از نظر بهره‌وری از DPSO ضعیف‌تر است. در حالت کلی، فقدان راهکاری که از نظر بهره‌وری، کارایی و توان در پوشش قوهٔ تعامل بالا را در آرایهٔ پوشش داشته باشد به چشم می‌خورد.

در این مقاله سعی شده است با تغییراتی در تولید دنبالهٔ آزمون با استفاده از الگوریتم بهینه‌سازی تودهٔ ذرات (PSO[5]) بهره‌وری و کارایی را تا حد قابل قبولی افزایش دهیم. این الگوریتم در بین الگوریتم‌های رایج تولید دنبالهٔ آزمون کمینه در آزمون ترکیبی بیشترین استفاده را دارد؛ زیرا از نظر سرعت و همچنین دقت از رقبای خود قوی‌تر است. شایان ذکر است که ما سایر الگوریتم‌های فرامکاشفه‌ای از جمله رقابت استعماری را پیاده‌سازی و نتایج آن را با الگوریتم‌های موجود مقایسه کرده‌ایم که در این بین الگوریتم PSO بسیار قوی‌تر است. یکی از ساختمان داده‌های مناسب برای ذخیره‌سازی نمونهٔ آزمون‌های پوشش داده نشده و محاسبهٔ وزن نمونهٔ آزمون در [۲] ارائه شده است که ما نیز در این پژوهش از این ساختار استفاده می‌کنیم. از طرفی، وزن یک نمونهٔ آزمون برابر تعداد حالات پوشش داده است که هر نمونهٔ آزمون وزن بیشتری داشته باشد، به دنبالهٔ آزمون اضافه می‌شود و اگر نمونهٔ آزمون‌ها برابر بود، با استفاده از راهکار استفاده‌شده در DPSO نمونهٔ آزمونی که اختلاف بیشتری با نمونهٔ آزمون‌های قبلی دارد، انتخاب می‌شود. این روال تا پوشش کامل ادامه دارد. اصلی‌ترین مشکل الگوریتم‌های فرامکاشفه‌ای در تولید آرایهٔ پوشش، گیر کردن در بهینهٔ محلی است که برای غلبه بر این مشکل در راهکار پیشنهادی در هر گام یک نمونهٔ آزمون تصادفی تولید و با بهترین مقدار مقایسه می‌شود و در صورت برتری جایگزین می‌گردد. در پایان بعد از تولید دنبالهٔ آزمون با استفاده از یک راهکار سادهٔ کمینه‌سازی تعداد نمونهٔ آزمون‌ها کاهش می‌یابد. اهداف و اقدامات کلی مقاله به شرح زیر است:

- استفاده از الگوریتم PSO برای تولید آزمون ترکیبی.
- بهره‌گیری از الگوریتم استفاده‌شده در DPSO برای کاهش نمونهٔ آزمون‌ها.
- استفاده از ساختمان دادهٔ استفاده‌شده در GS برای کاهش

1. In-Parameter-Order-General
2. Classification-Tree Editor eXtended Logics (CTE-XL)
3. Discrete Particle Swarm Optimization
4. Genetic Strategy
5. Particle Swarm Optimization

زمان تولید مجموعهٔ آزمون.

- استفاده از تابع کمینه‌ساز برای کاهش تعداد نمونهٔ آزمون‌ها.

ادامهٔ مقاله به این شرح است: در بخش دوم با عنوان ادبیات موضوعی سعی می‌شود مطالبی دربارهٔ آرایهٔ پوشش و الگوریتم PSO ارائه شود. بخش سوم به راهکارهای ارائه‌شده در حوزهٔ آرایهٔ پوشش اختصاص داده شده است. در این فصل، استراتژی‌های مبتی بر هوش مصنوعی و مبتنی بر محاسبات مورد بحث قرار می‌گیرند. بخش چهارم مراحل دقیق پیاده‌سازی راهکار پیشنهادی و تعیین پارامترهای الگوریتم PSO را تشریح می‌کند. بخش پنجم نتایج مقایسهٔ راهکار پیشنهادی با راهکارهای در دسترس را نشان می‌دهد. و بخش ششم نیز به نتیجه‌گیری و کارهای آتی اختصاص داده شده است.

۲. ادبیات موضوعی

۲. ۱. آرایهٔ پوشش

سیستم‌های تحت آزمون بزرگ، پارامترهای زیادی دارند و اغلب این پارامترها دارای مقادیر بزرگی هستند. برای آزمون کامل نیاز است تمام حالت‌های سیستم آزمون شود. برای مثال اگر یک سیستم دارای ۱۰ پارامتر باشد و هر پارامتر ۵ مقدار داشته باشد، می‌بایست تعداد 5^{10} نمونهٔ آزمون به سیستم داده شود که تعداد آزمون کامل صورت گیرد که این تعداد بسیار زیاد است و برای سیستم‌های بزرگ‌تر عملاً آزمون کامل غیرممکن می‌شود. آرایهٔ پوشش یک روش ریاضی در آزمون ترکیبی است که در ساده‌ترین شکل ممکن یک جدول است که هر سطر آن یک نمونهٔ آزمون است. هر پارامتر در آرایهٔ پوشش یک ستون از جدول است [۲]. آرایهٔ پوشش به دو شکل نمایش داده می‌شود: اگر مقادیر تمام پارامترها برابر باشند، به‌صورت $CA(N;\ t, v^p)$[1] نمایش داده می‌شود که p تعداد پارامترها، v مقدار پارامترها، t قوه تعامل (t-way) است ($2 \le t \le p$) و N تعداد نمونهٔ آزمون‌های انتخابی است که پوشش کامل را برای قوهٔ تعامل t به ارمغان می‌آورد. اما اگر مقادیر پارمترها یکسان نباشد، به‌صورت $CA(N;\ t, p, v)$ نشان داده می‌شود و در آن v برداری است که مقادیر پارامترها را به‌ترتیب در خود نگه می‌دارد. مسئلهٔ اصلی در آرایهٔ پوشش به حداقل رساندن N است؛ به‌عبارتی هرچه مقدار N کوچک‌تر باشد، استراتژی انتخاب نمونهٔ آزمون‌ها قوی‌تر است. برای تشریح بیشتر پوشش ارائه از مثال آژانس مسافرتی [۵] به‌عنوان یک مثال اجرایی استفاده می‌کنیم. این سیستم از پنج جزء تشکیل شده که عبارت است از: مشتری، آژانس مسافرتی، بانک، خط هوایی و هتل. ارتباط بین اجزا در شکل (۱) نشان داده شده است. هرکدام از این اجزا به‌عنوان یک پارامتر مستقل شناخته می‌شود و می‌توانند یک یا چند مقدار را بگیرند. جدول (۱) این پارامترها و مقادیرشان را نشان می‌دهد. پنج‌تایی (سجاد،رخش،آسمان،گلستان،ملت) یک نمونهٔ آزمون از سیستم را تشکیل می‌دهد. 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 نمونهٔ آزمون مشخص می‌شود. مجموعه‌ای از این نمونه آزمون‌ها یک دنبالهٔ آزمون را تشکیل می‌دهند. برای شناسایی خطا باید دنبالهٔ آزمون ما تمام نمونهٔ آزمون‌های ممکن را شامل شود.

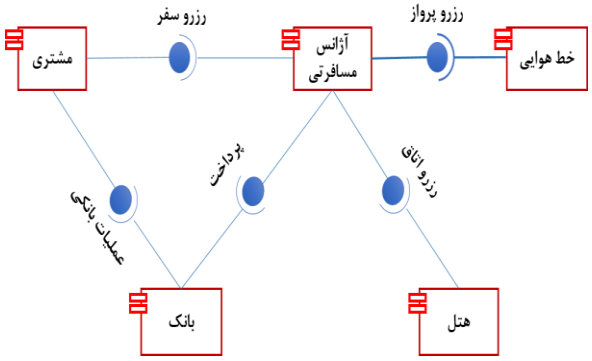
جدول ۱: مقادیر اجزای آژانس مسافرتی

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 مثال فرض خواهیم کرد که «سجاد» (مشتری ۱) اجازهٔ عملیات بانکی با بانک «ملت» را ندارد. پس زمانی که پارامتر مشتری، «سجاد» است و بانک «ملت» است، خطای سیستم رخ می‌دهد. برای شناسایی این خطا حالت دوتایی (سجاد،–،–،–،ملت) حداقل ۱ بار باید توسط دنبالهٔ آزمون پوشش داده شود. این حالت را می‌توان به شکل (۱،–،–،–،۲) نمایش داد. در آزمون کامل نرم‌افزار تمام حالت‌های ممکن از پارامترها باید پوشش داده شود. در مثال ما نیاز به تولید ۲×۲×۳×۲×۳=۷۲ نمونهٔ آزمون است. هزینهٔ آزمون زمانی که تعداد پارامترها و مقادیر آن‌ها زیاد باشد، بسیار بالا می‌رود.

1. Covering Array (CA)

علاوه بر این چون تعامل تعداد محدودی از پارامترها باعث بروز خطا می‌شود [۶] تولید آزمون کامل اصلاً مقرون‌به‌صرفه نیست و باید از تکنیک‌هایی برای حل این مسئله استفاده کرد که آن تکنیک آرایهٔ پوشش است.



شکل (۱): اجزای آژانس مسافرتی

بنابرین به‌جای آزمون کامل از آزمون t-way استفاده می‌کنیم که مقدار t در بازه $2 \leq t \leq p$ می‌باشد. مقدار t مشخص‌کنندهٔ این است که دنبالهٔ آزمون ما به‌جای اینکه تمام ترکیبات p را پوشش دهد. باید شامل تمام ترکیبات t تایی از پارامترها باشد. اگر مقدار t در تولید آزمون زیاد باشد، باعث به‌وجودآمدن تعداد زیادی نمونهٔ آزمون بی‌ارزش می‌شود و اگر تعداد t کم باشد، آن تعاملاتی را که باعث وقوع خطا می‌شوند، نمی‌توانیم شناسایی کنیم. در نتیجه انتخاب مناسب t بسیار حائز اهمیت است. همان طور که اشاره شد، تمام حالات مثال آژانس مسافرتی ۷۲ نمونهٔ آزمون است. با اعمال 2-way (t = 2) تعداد کل نمونهٔ آزمون‌ها ۹ تا می‌شود که 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 عبارت‌اند از: (–،–،پارسیان،پاسارگاد)، (–،–،گلستان،پاسارگاد)، (–،–،پارسیان،سپه)، (–،–،گلستان،سپه)، (–،–،پارسیان،ملت)، (–،–،گلستان،ملت)، تمام این حالت‌ها را در جدول (۲) می‌توان مشاهده کرد.

برای مثال دیگر، سیستمی با ۴ پارامتر ۲ مقداری را در نظر بگیرد. در حالت معمول، تعداد نمونهٔ آزمون برای این پیکربندی $2^4 = 16$ (v^p) است که در شکل (۳) نشان داده شده است. اما با اعمال پیکربندی CA (N; 2, 2^4)، تعداد نمونهٔ 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، دنبالهٔ آزمون تکمیل می‌شود. در حالت کلی، تعداد کل پوشش‌ها برای پیکربندی‌هایی که پارامترهایی با مقادیر برابر دارند، از رابطهٔ (۱) به دست می‌آید و اگر مقادیر برابر نباشند، از رابطهٔ (۲) استفاده می‌شود.

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

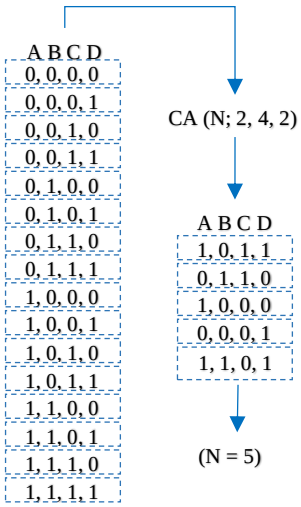
$$Max\_Coverage = \binom{p}{t} * |v_1| * |v_2| * \dots * |v_p| \qquad (۲)$$

جدول (۲): دنبالهٔ 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)

۲. ۲. الگوریتم PSO

یکی از شناخته‌شده‌ترین الگوریتم‌های مبتنی بر جمعیت، الگوریتم بهینه‌سازی تودهٔ ذرات (PSO) است [۷]. این الگوریتم، از رفتار حیواناتی که در توده (گروه) زندگی می‌کنند الهام می‌گیرد و در مسائل بهینه‌سازی کاربرد فراوانی دارد. در واقع، ذرات راه‌حل‌های مسئله را تشکیل می‌دهند و سعی می‌کنند به موقعیت بهتری در توده دست یابند. هر ذره، چهار ویژگی دارد که عبارت‌اند از: موقعیت جاری (Xi)، سرعت جاری (vi)، بهترین موقعیت فردی ($pBesti$) و بهترین موقعیت جمعی در توده ($gBesti$). لازم است هر ذره، ویژگی‌های خود را در فضای جست‌وجوی مسئله (مجموعه‌ای از راه‌حل‌ها) به‌روزرسانی کند. ازاین‌رو، این الگوریتم از تابعی به نام تابع هزینه استفاده کرده و جابه‌جایی ذرات را به این وسیله کنترل می‌کند. هر مسئله، تابع هزینهٔ خاص خود را دارد. یک ذره می‌تواند سرعت خود را طبق رابطهٔ (۳) به‌روز کند که مقدار به‌دست‌آمده در یافتن موقعیت جدید آن به کار می‌رود رابطهٔ (۴).



شکل (۲): نمایش پوشش ارائه با قوهٔ ثابت و متغیر

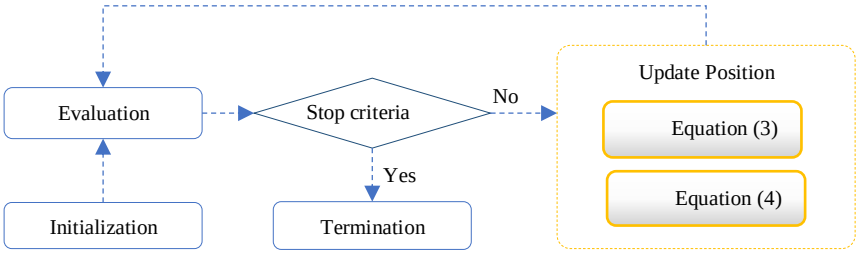
$$(t) = w(t-1) + cr_{jd}(pBest_{jd}(t-1) - x_{jd}(t-1)) + \acute{c}\acute{r}_{jd}(gBest_{jd}(t-1) - x_{jd}(t-1)) \quad (۳)$$

$$x_{jd}(t) = x_{jd}(t-1) + (t) \quad (۴)$$

در رابطهٔ (۳)، w وزن اینرسی است و تأثیر سابقهٔ سرعت را بر روی توانایی‌های فردی ($pBest$) و جمعی ($gBest$) ذرات کنترل می‌کند. c و $\acute{c}$ مقادیر ثابتی هستند که نرخ حرکت ذره به‌سمت بهترین موقعیت محلی و بهترین موقت سراسری را مشخص می‌کنند. این مقادیر معمولاً به شکلی انتخاب می‌شوند که مجموع آن‌ها از ۴ بیشتر نشود. در غیر این صورت، هر دو مقدار سرعت و موقعیت به‌سمت بی‌نهایت می‌روند r و $\acute{r}$ هم اعداد تصادفی در بازه [۰ و ۱] هستند و از توزیع یکنواخت به دست می‌آیند. بردار سرعت و مقادیر موقعیت در هر تکرار محاسبه می‌شوند تا جواب بهینه حاصل شود. مراحل اجرای الگوریتم PSO در شکل (۳) نشان داده شده است.

۳. کارهای مرتبط

به‌طور کلی، استراتژی‌های تولید آزمون ترکیبی به دو دستهٔ ریاضی و محاسباتی تقسیم می‌شوند [۸]. روش ریاضی از ساختار ارائهٔ متعامد OA (Orthogonal Array) در تولید آزمون ترکیبی استفاده می‌کند. این روش فقط قادر به تولید دنبالهٔ آزمون برای پیکربندی‌های خاص استفاده می‌شود. Combinatorial Test Services و TConfig دو استراتژی شناخته‌شده در روش‌های ریاضی هستند. اما عمده پژوهش در دنیای آزمون، ترکیباتی از روش‌های محاسباتی استفاده می‌کند. این روش به‌طور کلی به دو دستهٔ استراتژی‌های محاسبات محض و هوش مصنوعی تقسیم می‌شوند که در ادامه به تشریح آن پرداخته می‌شود.



شکل (۳): فلوچارت الگوریتم PSO

۳. ۱. استراتژی‌های محاسباتی محض

به‌طور روش‌های محاسباتی برای ساخت آرایهٔ پوشش از دو مکانیزم یک-آزمون-در-یک-زمان (OTAT)[1] و یک-پارامتر-در-یک-زمان (OPAT)[2] استفاده می‌کند. در روش OTAT در هر گام یک سطر (نمونهٔ آزمون) به دنبالهٔ آزمون اضافه می‌شود تا زمانی که پوشش کامل شود، اما در روش OPAT در هر گام یک پارامتر به دنبالهٔ آزمون اضافه می‌شود و زمانی که پوشش کامل شد، یک پارامتر دیگر به دنبالهٔ آزمون اضافه می‌شود و همین روال تا تکمیل شدن پارامترها و پوشش کامل ادامه می‌یابد. استراتژی‌های شناخته‌شده با روش OTAT عبارت‌اند از: AETG [۹]، mAETG [۱۰]، PICT [۱۱]، DDA [۱۲] [۱۳]، CTE-XL [۱۴ و ۱۵]، TVG [۱۶ و ۱۷]، Jenny [۱۸]، و ITCH [۱۹] است. استراتژی AETG اولین استراتژی مبتنی بر OTAT بود که به‌صورت حریصانه نمونهٔ آزمون‌ها را انتخاب و به دنبالهٔ آزمون اضافه می‌کرد تا پوشش کامل شود. این استراتژی بعدها در mAETG و mAETG-sat توسعه داده شد. دیگر استراتژی مبتنی بر OTAT که به‌صورت حریصانه نمونهٔ آزمون را انتخاب می‌کرد، PICT است. این استراتژی Random Base است و معمولاً نتایج مطلوبی را از نظر بهره‌وری ندارد، اما کارایی مناسبی دارد. TVG نیز دیگر استراتژی از خانوادهٔ OTAT است که مانند PICT در دسترس می‌باشد. این استراتژی از سه الگوریتم T-Reduce، plus-one و Random Sets برای تولید دنبالهٔ آزمون استفاده می‌کند که T-Reduce نتایج بهتری از رقبای خود دارد. Jenny دیگر استراتژی در دسترس و مبتنی بر OATA است که از نظر بهره‌وری نتایج قابل قبولی دارد و قادر است قوهٔ تعاملات بالا را نیز پشتیبانی کند. می‌توان گفت قدرتمندترین استراتژی مبتنی بر محاسبات محض از نظر بهره‌وری ITCH است. دنبالهٔ آزمون در این روش با جست‌وجوی جامع (exhaustive) تولید می‌شود. این استراتژی در برخی از پیکربندی‌ها نتایجی حتی بهتر از استراتژی‌های مبتنی بر هوش مصنوعی از لحاظ بهره‌وری دارد اما از نظر کارایی ضعیف است و فقط تا t=4 توان تولید دنبالهٔ آزمون را دارد. در بین استراتژی‌های مذکور تنها استراتژی که دنبالهٔ آزمون را با Classification-Tree Method (CTM) تولید می‌کند، CTE_XL است. این استراتژی اغلب تا t=3 دنبالهٔ آزمون تولید می‌کند و از نظر کارایی و بهره‌وری نیز نتایج مناسبی ندارد. آخرین استراتژی محاسباتی محض مبتنی بر OTAT که در گفتار مورد ارزیابی قرار می‌گیرد، GTWay است. GTWay نتایج مناسبی از نظر کارایی و بهروری دارد. توان تولید دنبالهٔ آزمون تا t = 12 را دارد. IPOG [۲۰]، IPOG-s [۲۱] و IPOG-D [۲۲] استراتژی‌های رایجی هستند که بر اساس OPAT گسترش داده شده‌اند. دو استراتژی IPOG و IPOG-D در ابزار ACTS [۲۳] ارائه شده‌اند و از نظر کارایی بهترین عملکرد را در تولید آرایهٔ پوشش دارند.

۳. ۲. استراتژی‌های مبتنی بر هوش مصنوعی

محبوب‌ترین روش در تولید آرایهٔ پوشش الگوریتم‌های مبتنی بر هوش مصنوعی هستند که بیشتر آن‌ها از روش OTAT برای گسترش دنبالهٔ آزمون استفاده می‌کنند. تعداد فراوانی از الگوریتم‌های هوش مصنوعی برای تولید آرایهٔ پوشش مورد استفاده قرار گرفته‌اند که برخی از آن‌ها که در مجلات و کنفرانس‌های معتبر چاپ شده، عبارت‌اند از: SA [۱۰ و ۲۴]، TS [۲۴]، GA [۲۴ و ۲]، ACA [۲۵]، PSO [۲۶ و ۴]، CS [۲۷] و HSS [۸] هستند. تولید آرایهٔ پوشش با الگوریتم‌های هوش مصنوعی را ابتدا استاردم [۲۴] با سه الگوریتم SA، GA و TS مطرح کرد. این استراتژی‌ها فقط توان تولید تعامل 2-way را داشتند. در بین این سه راهکار، SA نتایج بهتری را از دو استراتژی GA و TS داشت همچنین GA نیز از TS بهتر بود. توسعهٔ روزافزون سیستم‌های نرم‌افزاری لزوم استفاده از تعاملات بالاتر را بیشتر احساس می‌کرد. در همین راستا کوهن در [۱۰]، SA و شیبا در [۲۵]، GA و ACA را برای پشتبانی از تعاملات تا t = 3 ارائه دادند. این استراتژی‌ها نتایج بسیار بهتری را از استراتژی‌ها مبتنی بر محاسبات داشتند. این استراتژی‌ها از الگوریتم‌های پیچیده برای کاهش تعداد نمونهٔ

1. One-test-at-a-time
2. One-parameter-at-a-time

آزمون استفاده می‌کردند و در نتیجه سرعت تولید کاهش می‌یافت و توان پشتیبانی از قوهٔ تعاملات بالا را نداشتند. استراتژی PSTG با حذف این الگوریتم‌های پیچیده و همچنین ارائهٔ یک ساختمان داده برای ذخیرهٔ حالت‌های پوشش داده نشده و سرعت بخشیدن به محاسبهٔ وزن نمونهٔ آزمون‌ها توانست تا t=6 را پشتیبانی کند. به نظر می‌رسد PSTG برای هر ترکیب یک ماترس در نظر گرفته بود که در زمان محاسبهٔ وزن نمونهٔ آزمون با استفاده از ایندکس آن ماتریس، پوشش یا عدم پوشش نمونهٔ آزمون را تشخیص می‌داد. استراتژی CS با تغییر در ساختمان داده PSTG توانست کارایی تولید آرایهٔ پوشش را بالا ببرد اما از نظر بهره‌وری برتری قابل قبولی نسبت به PSTG را ارائه نداد و تا t=6 را پشتیبانی می‌کرد. دیگر استراتژی قدرتمند در تولید آرایهٔ پوشش، HSS مبتنی بر Harmony Search Algorithm است. این استراتژی توان تولید تا t=15 را پشتیبانی می‌کند و از نظر بهره‌وری نتایج به‌مراتب بهتری را از PSTG دارد اما در خصوص کارایی نتایج آن مورد ارزیابی قرار نگرفته است. مشکل اصلی در تولید آرایهٔ پوشش با الگوریتم‌های مبتنی بر تودهٔ ذرات این است که اعمال velocity لزوماً نتیجه را بهتر نخواهد کرد و حتی در برخی از موارد نتیجهٔ عکس دارد. استراتژی DPSO برای غلبه بر این مشکل راهکار بسیار کارآمدی را ارائه داد. این استراتژی بهترین نتایج را در بین استراتژی‌های موجود از نظر بهره‌وری داشت اما از نظر کارایی مناسب نبود. ساختار پیاده‌سازی این استراتژی توان پشتیبانی قوه‌های بالا را هم دارد، اما با در نظر گرفتن مدت زمان یک روزه برای هر پیکربندی این استراتژی تا t=10 را پشتیبانی می‌کند [۲]. دیگر استراتژی که در [۴] ارائه شد، CPSO بود که این استراتژی از نظر کارایی از DPSO قوی‌تر بود اما بهره‌وری DPSO قدرت بیشتری دارد. این دو استراتژی در دسترس هستند [۲۸]. استراتژی GS [۲] نیز یکی دیگر از استراتژی‌هایی است که CA را پشتیبانی می‌کند. این استراتژی با بهره‌گیری از ساختار بیتی و تغییر در الگوریتم ژنتیک نتایج مناسبی از نظر زمان نسبت به استراتژی‌های مبتنی بر هوش مصنوعی دارد و قادر است تا t=20 را پشتیبانی کند.

۴. پیاده‌سازی

در این پژوهش برای تولید آرایهٔ پوشش از الگوریتم قدرتمند PSO استفاده شده است. ما در این پژوهش بیش از ۱۰ الگوریتم فرامکاشفه‌ای را پیاده‌سازی کرده‌ایم که در این بین الگوریتم PSO از نظر کارایی و بهره‌وری نتایج مناسب‌تری را تولید می‌کرد. گام‌های پیاده‌سازی را در ادامه شرح می‌دهیم.

گام ۱: ایجاد جمعیت اولیه

در این مرحله، مقداردهی اولیهٔ پارامترهای مربوط به الگوریتم PSO تعیین می‌شود و همچنین جمعیت اولیه به‌صورت تصادفی ایجاد می‌گردد. هر ذره در الگوریتم تودهٔ ذرات معادل یک نمونهٔ آزمون است. این ذره شامل به تعداد پارامترهای پیکربندی دارای سلول است که هرکدام از این سلول‌ها مقداری بین 0 تا $v-1$ را اختیار می‌کنند.

گام ۲: ساخت ماتریس پوشش

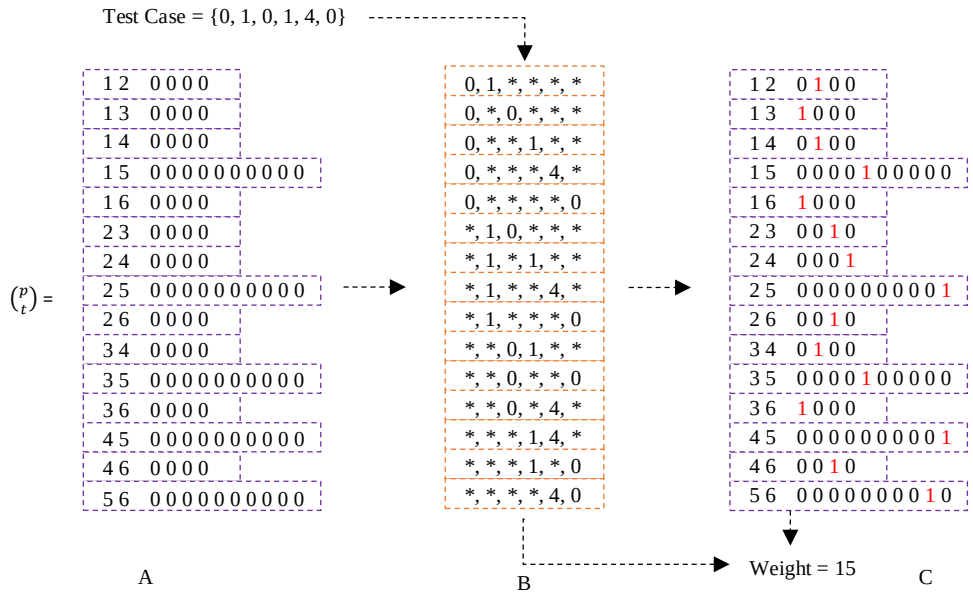
یک عامل مهم در زمان تولید آرایهٔ پوشش، محاسبهٔ وزن نمونهٔ آزمون است. برای محاسبهٔ وزن یک نمونهٔ آزمون می‌بایست در ابتدا تمام حالت‌های ممکن که برای پوشش کامل نیاز است (رابطهٔ ۱ و ۲) ذخیره شود. ساختمان داده‌های مختلفی برای این منظور ارائه شده است که ما در این پژوهش از ساختمان دادهٔ استفاده‌شده در [۲] استفاده می‌کنیم. تعداد سطرهای این ساختمان داده C_t^p است و هر سطر دارای دو بخش است. بخش اول شمارهٔ ترکیب را ذخیره می‌کند و بخش دوم دارای $|v_1|*|v_2|*\dots*|v_t|$ سلول است که در حالت پیش‌فرض، مقدار ۰ به معنای عدم پوشش را دارد. برای تشریح کامل‌تر، یک مثال را ارائه می‌دهیم: فرض کنید یک سیستم با ۶ پارامتر که هرکدام به‌ترتیب ۲، ۲، ۲، ۵ و ۲ مقدار را داشته باشند و فرض می‌کنیم t=2 است. در قسمت اول، در سطر اول مقدار ۱ و ۲ (ترکیب ۱ و ۲) قرار می‌گیرد و تعداد سلول‌های قسمت دوم نیز برابر $|v_1|*|v_2| = 2*2 = 4$ است. سطر دوم و سوم نیز شرایط مشابهی دارد، با این تفاوت که در قسمت اول به‌ترتیب مقادیر (۱ و ۳) و (۱ و ۴) به معنای ترکیب پارامتر اول و سوم و ترکیب پارامتر اول و چهارم است. قسمت اول سطر چهارم (۱ و ۵) است و قسمت دوم آن برابر $|v_1|*|v_5| = 2*$

$10 = 5$ است. سایر سطرهای ماتریس نیز به همین منوال ایجاد می‌شود. شکل (a۴) پوشش را برای مثال فوق نشان می‌دهد. برای محاسبهٔ وزن یک نمونهٔ آزمون سطر به سطر ماتریس پوشش باید کاوش شود. با استفاده از قسمت اول هر سطر مقادیر لازم از نمونهٔ آزمون استخراج و معادل دهدهی آن محاسبه می‌شود، سپس سلول متناظر چک می‌شود. اگر این مقدار برابر با صفر باشد، مقدار سلول به یک تغییر می‌کند و یک واحد به وزن اضافه می‌شود. این روال تا آخرین سطر ادامه می‌یابد. مراحل محاسبهٔ وزن یک نمونهٔ آزمون را با یک مثال شرح می‌دهیم. برای سیستم دارای شش پارامتر فوق نمونهٔ آزمون (۰،۱،۴،۰،۱،۰) را در نظر می‌گیریم. در سطر اول مقدار ۱ و ۲ ذخیره شده است که با توجه به این دو مقدار ما سلول اول و دوم نمونهٔ آزمون یعنی (۰،۱) را انتخاب می‌کنیم (شکل b۴) و معادل دهدهی آن را محاسبه می‌کنیم، بعد از آن سلول متناظر با آن را (سومین سلول) را که مقدار صفر دارد به یک تغییر می‌کند و یک واحد را نیز به وزن اضافه می‌کنیم (شکل c۴). هر سطر حداکثر یک پوشش را به همراه دارد در حالت اولیه با توجه به اینکه هیچ حالت پوشش داده شده وجود ندارد، تمام ترکیبات نمونهٔ آزمون یک پوشش را در بر دارد؛ لذا اولین نمونهٔ آزمون را می‌توان به‌طور تصادفی به دنبالهٔ آزمون اضافه کرد. حداکثر وزن یک نمونهٔ آزمون برابر با تعداد سطرهای ماتریس پوشش است

در نتیجه، اولین نمونهٔ آزمون وزنی برابر با C_t^p دارد. هرچه وزن یک نمونهٔ آزمون بیشتر باشد، ارزش آن نمونهٔ آزمون بیشتر است. اما نمونهٔ آزمون‌های با وزن برابر می‌توانند ارزش یکسان نداشته باشند و مثالی که در [۴] آورده شده است صحت این گفتار را تأیید می‌کند. این مثال برای پیکربندی CA (N; 2, 3^4) ارائه و در جدول (۳) نشان داده شده است. همان طور که ملاحظه می‌شود، دو دنبالهٔ آزمون برای پیکربندی مذکور آورده شده است. در دنبالهٔ آزمون اولی (Sequence 1) سه نمونهٔ آزمون ابتدایی دارای وزنی برابر ۶ هستند و سه نمونهٔ آزمون Sequence 2 نیز همین وضعیت را دارند. اما تفاوت در انتخاب نمونهٔ آزمون بعدی است. در Sequence 1 دیگر هیچ نمونهٔ آزمونی با وزن ۶ وجود ندارد، اما در Sequence 2 نمونهٔ آزمون (۲ ۱ ۰ ۱) وزنی برابر ۶ دارد. در حالت کلی برابر انتخاب نمونهٔ آزمون‌های با وزن مساوی، بهتر است نمونهٔ آزمونی انتخاب شود که کمترین اختلاف را با نمونهٔ آزمون قبلی خود داشته باشد.

جدول (۳): دنباله‌های مختلف نمونهٔ آزمون

#	Sequence 1	Sequence 2
1	0 0 0 0	0 0 0 0
2	1 1 1 1	0 1 1 1
3	2 2 2 2	0 2 2 2
4	-	1 0 1 2



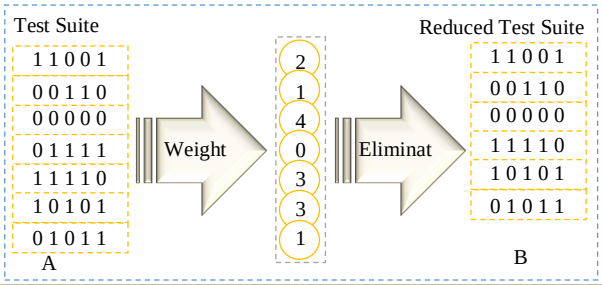
شکل (۴): مراحل محاسبهٔ وزن یک نمونهٔ آزمون

گام ۳: تولید CA با PSO

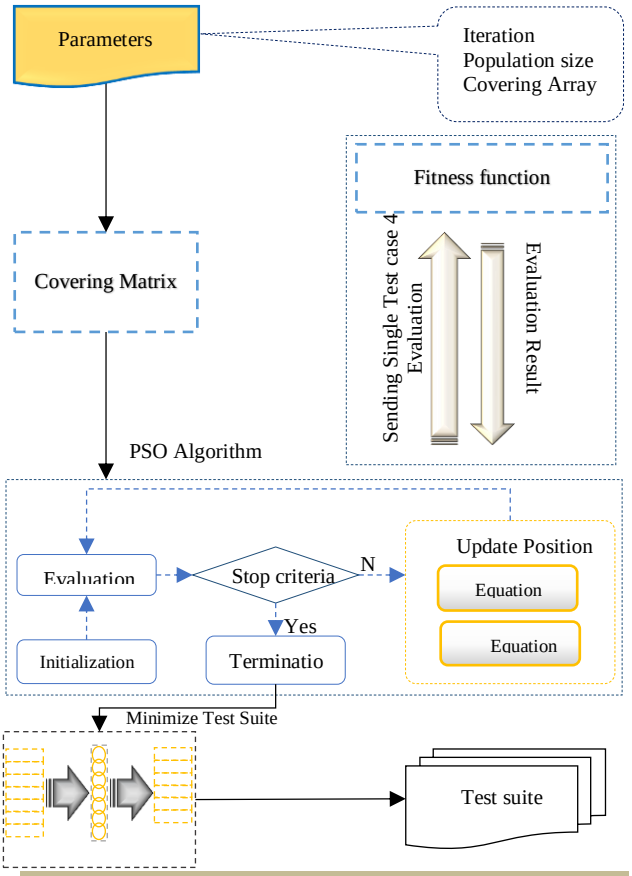
بعد از ایجاد جمعیت اولیه در الگوریتم PSO، نمونهٔ آزمون با بیشترین وزن انتخاب (best) و این نمونهٔ آزمون جایگزین gbest می‌شود و روال کامل الگوریتم PSO بر روی این نمونهٔ آزمون انجام می‌گیرد و یک نمونهٔ آزمون تغییریافته را برمی‌گرداند؛ اگر نتیجه بهتر شد، best به‌روز می‌شود. در ادامه best را با یک نمونهٔ آزمون تصادفی مقایسه می‌کنیم، اگر بهبود حاصل شد با best جایگزین می‌شود، و این بهترین آزمون به دنبالهٔ آزمون نهایی اضافه می‌شود. نمونهٔ آزمون اضافه‌شده به دنبالهٔ آزمون قطعاً دارای وزن مثبت بوده، اما در ادامه ممکن است با اضافه شدن نمونهٔ آزمون‌های بعدی ارزش این نمونهٔ آزمون صفر شود. این نمونهٔ آزمون را نمونهٔ آزمون حشو می‌نامیم. در این پژوهش با ارائهٔ یک راهکار جدید نمونهٔ آزمون‌های حشو را شناسایی و حذف می‌کنیم و دنبالهٔ آزمون مینی‌مایز می‌شود. در ادامه به شرح این تابع پرداخته می‌شود.

گام ۴: کمینه‌کردن دنبالهٔ آزمون

در راهکارهای مبتنی بر OTAT در هر گام یک نمونهٔ آزمون که معمولاً این نمونهٔ آزمون بیشترین تعداد پوشش را دارد، به دنبالهٔ آزمون نهایی اضافه می‌شود. نمونهٔ آزمون انتخابی علاوه بر پوشش جدید حالت‌های پوشش داده شده نیز را در بر دارد. با افزایش تعداد نمونهٔ آزمون‌ها ممکن است تمام حالت‌هایی که یک نمونهٔ آزمون پوشش داده شده است، توسط دیگر نمونهٔ آزمون‌ها پوشش داده شود که در این حالت این نمونهٔ آزمون اضافی است و باید پاک شود. برای فهم دقیق‌تر دنبالهٔ آزمون، شکل (۵) را در نظر بگیرید. ابتدا می‌بایست مجدداً وزن نمونهٔ آزمون‌ها محاسبه شود اما در اینجا منظور از وزن تعداد حالتی است که نمونهٔ آزمون به‌تنهایی توانسته است آن را پوشش دهد. برای مثال سطر یک دارای مقدار (*,*,0,0,*) است که در سطر چهار نیز مشاهده می‌شود پس وزنی را برای آن در نظر نمی‌گیریم. اما در همین سطر مقادیر (*,*,0,*,1) و (*,1,*,0,*) وجود دارند که این مقادیر در هیچ یک از نمونهٔ آزمون‌ها ظاهر نشده است؛ در نتیجه وزن این نمونهٔ آزمون ۲ است و قابل حذف نیست. اما تمام ۶ حالت سطر ۴ توسط نمونهٔ آزمون‌های دیگر پوشش داده می‌شود و وزن این نمونهٔ آزمون در عمل صفر می‌شود و می‌توان آن را حذف کرد (شکل 5B). مراحل کلی الگوریتم پیشنهادی در شکل (۶) نشان داده شده است.



شکل (۵): تابع حذف نمونهٔ آزمون‌های حشو

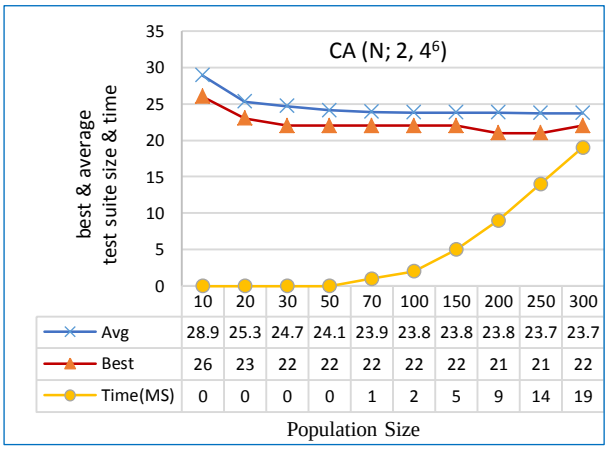


شکل (۶): مراحل اجرای الگوریتم پیشنهادی

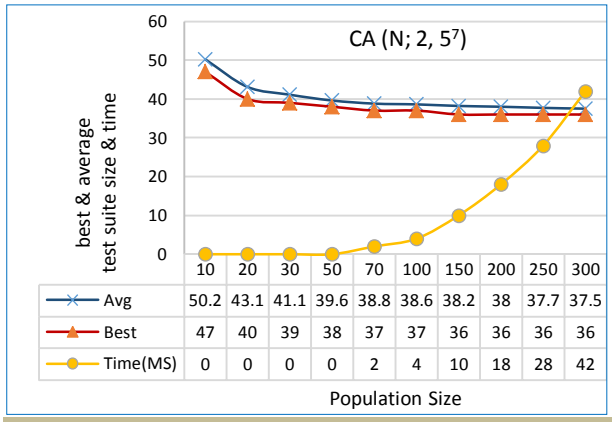
۵. تنظیم پارامترها

یکی از مباحث مبهم در الگوریتم‌های فرامکاشفه‌ای انتخاب دقیق پارامترهای آن است. الگوریتم PSO دارای ۵ پارامتر C1, C2, w و repetition است. تأثیر تغییر سه پارامتر C1, C2 و w مانند [۲۶] است و ما برای جلوگیری از افزایش حجم مقاله از

تشریح و بررسی آن خودداری می‌کنیم؛ اصلی‌ترین پارامتر تعیین‌کننده در تولید بهینهٔ آرایهٔ پوشش، اندازهٔ جمعیت (popsize) است. با توجه به اینکه فضای جست‌وجو به اندازهٔ پیکربندی‌ها بستگی دارد، انتخاب مقدار popsize رابطهٔ مستقیم با اندازهٔ پیکربندی دارد و هرچه پیکربندی بزرگ‌تر باشد، مقدار popsize نیز باید بیشتر باشد. در ادامه با افزایش popsize تأثیر آن را بر روی پیکربندی CA (N; 2, 4^6) مشاهده می‌کنیم. نتایج این بررسی در شکل (۷) نشان داده شده است. همان طور که ملاحظه می‌شود، بهترین مقدار با دنبالهٔ آزمون با ۲۱ نمونهٔ آزمون است که popsize=200 به دست می‌آید؛ اما میانگین کماکان تا جمعیت ۳۰۰ در حال بهبود است و همچنین با افزایش جمعیت زمان تولید دنبالهٔ آزمون نیز افزایش می‌یابد. زمان برحسب میلی ثانیه است. در ادامهٔ بررسی، پیکربندی CA (N; 2, 5^7) را در نظر می‌گیریم. این بار، بازهٔ تغییرات popsize نیز بین ۱۰ تا ۳۰۰ است. این اعداد ممکن است برای برخی از پیکربندی‌های بزرگ یکسان نباشد اما جمعیت بیشتر از ۳۰۰ سرعت را به‌شدت کم می‌کند و همچنین جمعیت کمتر از ۱۰ نیز بهره‌وری را پایین می‌آورد. با وجود این ممکن است پیکربندی وجود داشته باشد که به جمعیت خارج از این بازه نیاز داشته باشد. همان طور که در شکل (۸) مشاهده می‌شود افزایش جمعیت تأثیر زیادی بر زمان اجرای الگوریتم دارد. بهترین مقدار در این پیکربندی در popsize=150 رخ می‌دهد اما برای جمعیت‌های بیشتر میانگین بهتر می‌شود.



شکل (۷): تأثیر افزایش جمعیت بر تعداد و زمان تولید دنبالهٔ آزمون



شکل (۸): تأثیر افزایش جمعیت بر تعداد و زمان تولید دنبالهٔ آزمون

۶. ارزیابی

ارزیابی در این پژوهش به سه دسته بهره‌وری (اندازه آرایه)، کارایی (زمان تولید) و قوه تعامل تقسیم می‌شود [۲]. بهره‌وری در واقع به تعداد نمونه آزمون‌های یک مجموعه آزمون اشاره می‌کند و هرچه این نمونه آزمون‌ها کمتر باشند بهره‌وری الگوریتم مربوطه قوی‌تر است. از طرف دیگر کارایی، زمان کل تولید مجموعه آزمون را بیان می‌کند که هرچه زمان تولید کمتر باشد الگوریتم قوی‌تر خواهد بود. بهره‌وری فقط به مراحل اجرای استراتژی بستگی دارد، در حالی که کارایی رابطه مسقیم با بستر سخت‌افزاری و نرم‌افزاری دارد [۲۷]. تعداد زیادی استراتژی برای تولید آرایه پوشش وجود دارد که بررسی کلیه آنها در این بخش امکان‌پذیر نیست، لذا ما سعی کردیم برای عادلانه نشان دادن مقایسات، تعدادی از استراتژی‌های در دسترس را مورد ارزیابی قرار دهیم. در بین استراتژی‌های هوش مصنوعی سه استراتژی CPSO، DPSO و GS انتخاب شده است. که DPSO از نظر بهره‌وری بهترین نتایج را در بین استراتژی‌های در دسترس دارد. در بین استراتژی‌های غیر هوش مصنوعی نیز Jenny، TConfig، PIC و IPOG انتخاب شده‌اند که تفاوت آن‌ها در بخش کارهای مرتبط ارائه شده است.

مشخصات بستر سخت‌افزاری عبارتند از: Windows 7 ، core™ i7QM CPU 2.20GHz و 6GB RAM و در محیط eclipse (jdk 1.8) کد نویسی آن صورت گرفته است.

استراتژی‌های غیر از هوش مصنوعی معمولاً قطعی هستند و اجرای چندبارهٔ آن‌ها تأثیری در نتیجه نخواهد داشت؛ اما استراتژی‌های مبتنی بر هوش مصنوعی غیرقطعی هستند و با اجرای مجدد نتیجه تغییر خواهد کرد. لذا ما هر پیکربندی را صد بار اجرا کرده و دو مقدار Best و Average که به ترتیب بیانگر بهترین و میانگین صد بار اجرا برای هر پیکربندی است. مقادیر مربوط به بهره‌وری برای استراتژی‌های GS، CPSO و DPSO در صورت موجود بودن مستقیماً از مقالات مربوط گرفته شده است، در غیر این صورت به تعداد برابر با الگوریتم پیشنهادی اجرا و نتیجه ثبت شده است. شایان ذکر است که برای تعاملات بالا تعداد دفعات تکرار کاهش یافته است. پارامترهای راهکار پیشنهادی در جدول (۴) نشان داده شده است.

جدول (۴): مقداردهی پارامترهای الگوریتم PSO

Algorithm	Parameter	Value
PSO	Repetition	10-20
	W	0.8
	C1	2
	C2	3

۳. ۱. بهره‌وری

مهم‌ترین معیار سنجش قدریت یک استراتژی در تولید آرایهٔ پوشش، بهره‌وری آن استراتژی است. در این قسمت با چهار جدول (۵) تا (۸) نتایج مقایسهٔ راهکار پیشنهادی را با سایر استراتژی‌ها نشان می‌دهیم. اولین ارزیابی را به t=2 اختصاص داده و نتایج مربوط به این ارزیابی را در جدول (۵) نشان می‌دهیم. همان طور که ملاحظه می‌شود، در این ارزیابی ۱۳ پیکربندی در نظر گرفته شده است که در این بین، استراتژی Jenny در یک مورد، CPSO در چهار مورد، DPSO در هشت مورد، GS و NEWPSO در یازده مورد بهترین خروجی را دارند. دومین ارزیابی مربوط به t=3 است. برای این ارزیابی ۱۴ پیکربندی در نظر گرفته شده که نتایج ارزیابی در جدول (۶) نشان داده شده است. در این ارزیابی به‌طور کلی CPSO چهار مورد، DPSO ۹ مورد، GS ۶ مورد و NEWPSO در ۱۲ مورد بهترین نتایج را تولید می‌کنند. جدول (۷) مربوط به مقایسهٔ استراتژی پیشنهادی با سایر استراتژی‌های دیگر در t=4 است. در این مقایسه هم مشخص است که قدرت استراتژی‌های مبتنی بر هوش مصنوعی از محاسبات بیشتر است. در این مقایسه نیز استراتژی پیشنهادی در ۵ پیکربندی بهترین عملکرد دارد. آخرین ارزیابی مربوط به efficiency برای t>4 است که نتایج آن در جدول (۸) نشان داده است. در این مقایسه، TConfig فقط در پیکربندی‌های CA (N; 5, 3^7) و CA (N; 6, 3^8) قادر به تولید دنبالهٔ آزمون در کمتر از ۲۴ ساعت است و در سایر موارد زمانی بیش از این را نیاز دارد و در جدول (۸) با ">day" مشخص می‌شود. استراتژی IPOG نیز توان تولید تا t=6 را دارد، اما مقادیر بزرگ‌تر از ۶ را پشتیبانی نمی‌کند. استراتژی DPSO نیز در ۱ مورد زمانی بیش از یک روز نیاز دارد. اما سایر استراتژی تعاملات بالا را پشتیبانی می‌کند؛ که در این بین استراتژی پیشنهادی از سایر استراتژی‌ها قوی‌تر است.

جدول (۵): مقایسهٔ بهره‌وری استراتژی‌ها در t=2

	Jenny	CPSO	DPSO	GS	NEWPSO
	N	N.Best	N.Best	N.Best	N.Best
CA (N; 2, 2^7)	8	7	7	6	6
CA (N; 2, 3^3)	9	9	9	9	9
CA (N; 2, 3^4)	13	9	9	9	9
CA (N; 2, 3^5)	14	11	11	11	11
CA (N; 2, 3^6)	15	14	14	13	13
CA (N; 2, 3^7)	16	15	15	14	14
CA (N; 2, 3^8)	17	15	15	15	15
CA (N; 2, 3^9)	18	16	15	15	15
CA (N; 2, 3^{10})	19	16	16	16	16
CA (N; 2, 3^{11})	17	16	17	16	16
CA (N; 2, 3^{12})	19	17	16	16	16
CA (N; 2, 4^7)	28	25	24	24	24
CA (N; 2, 5^7)	37	36	34	36	36

۳. ۲. کارایی

مقایسهٔ بعدی مربوط به زمان اجرای تولید CA است. این بخش از ارزیابی را به مقایسهٔ راهکار پیشنهادی با راهکارهای مبتنی بر هوش مصنوعی اختصاص می‌دهیم. همان طور که قبلاً اشاره شد، یکی از عوامل تأثیرگذار در زمان تولید CA در راهکار پیشنهادی تعیین پارامترهای الگوریتم است. برای این مقایسه برای popsize، نرخ تقاطع و نرخ جهش به‌ترتیب مقادیر ۵۰، ۰/۸ و ۰/۸ را در نظر گرفته‌ایم. در این مقایسه، زمان تولید یک نمونهٔ آزمون در نظر گرفته می‌شود. برای مثال

DPSO در زمان ۳۰ میلی‌ثانیه موفق به تولید دنبالهٔ آزمون با ۱۴ نمونهٔ آزمون می‌شود لذا زمان لازم برای یک نمونهٔ آزمون ۲/۱۴ (۱۴/۳۰) است. نتایج مربوط به این مقایسه در جدول (۹) نشان داده است. همان طور که ملاحظه می‌شود، استراتژی پیشنهادی بسیار قوی‌تر از استراتژی‌های مبتنی بر هوش مصنوعی است.

جدول (۶): مقایسهٔ بهره‌وری استراتژی‌ها در t=3

	Jenny N	CPSO N.Best	DPSO N.Best	GS N.Best	NEWPSO N.Best
CA (N; 3, 2^7)	14	12	15	12	12
CA (N; 3, 2^8)	14	16	16	14	14
CA (N; 3, 2^9)	17	16	16	16	16
CA (N; 3, 3^4)	34	30	28	27	27
CA (N; 3, 3^5)	40	38	41	38	38
CA (N; 3, 2^{10})	18	16	16	16	16
CA (N; 3, 3^6)	51	42	33	43	42
CA (N; 3, 3^7)	51	49	48	49	48
CA (N; 3, 3^8)	58	53	52	54	52
CA (N; 3, 3^9)	62	58	56	58	56
CA (N; 3, 3^{10})	65	61	59	61	59
CA (N; 3, 3^{11})	65	63	63	63	63
CA (N; 3, 3^{12})	68	68	65	67	66
CA (N; 3, 4^7)	124	115	112	116	114

جدول (۷): مقایسهٔ بهره‌وری استراتژی‌ها در t=4

	Jenny N	CPSO N.Best	DPSO N.Best	GS N.Best	NEWPSO N.Best
CA (N; 4, 2^7)	31	24	31	27	25
CA (N; 4, 2^8)	37	32	32	30	30
CA (N; 4, 2^9)	37	33	34	33	25
CA (N; 4, 2^{10})	39	37	34	25	28
CA (N; 4, 3^5)	109	94	81	88	90
CA (N; 4, 3^6)	140	132	131	129	130
CA (N; 4, 3^7)	169	153	150	152	153
CA (N; 4, 3^8)	187	174	171	171	171
CA (N; 4, 3^9)	206	191	187	187	192
CA (N; 4, 3^{10})	221	211	206	206	206
CA (N; 4, 3^{11})	236	226	221	223	221
CA (N; 4, 3^{12})	252	242	237	236	238

یکی از روش‌های ارزیابی کارایی استراتژی‌ها استفاده از روش آماری Wilcoxon signed rank sum است. این آزمون بزرگیِ اختلاف میان جفت‌ها را در نظر می‌گیرد؛ به‌عبارت دیگر، این آزمون برای بررسی اختلاف معنایی بین دو دسته از اعداد به کار می‌رود. از ابزار SPSS می‌توان برای استفاده از این آزمون استفاده کرد. این ارزیابی دو خروجی Test Statistics و ranks است. برای دو استراتژی A و B ranks تعداد نمونه‌های A>B، A<B و A=B را نشان می‌دهد و قسمت Test Statistics دو مقدار z و Asymp. Sig. (2-tailed) دارد که تشریح z در این مقاله نمی‌گنجد و مقدار دیگر تعیین‌کنندهٔ تفاوت معنادار است که اگر این مقدار کمتر از ۰/۰۵ باشد به این معناست که تفاوت معنا داری بین دو استراتژی وجود دارد. نتایج مربوط به آزمون ویلکاکسون جدول (۵) و (۹) در جدول (۱۰) نشان داده شده است.

جدول (۸): مقایسهٔ بهره‌وری استراتژی‌ها در t > 4

	Jenny N	CPSO N.Best	DPSO N.Best	GS N.Best	NEWPSO N.Best
CA (N; 5, 3^7)	458	441	428	431	433
CA (N; 6, 3^8)	1466	1397	1402	1398	1396
CA (N; 7, 3^9)	4746	4422	4427	4437	4428
CA (N; 8, 3^{10})	14999	13925	13933	13907	13907
CA (N; 9, 3^{11})	47009	43587	>day	43808	43585
CA (N; 10, 3^{12})	147004	135498	>day	136096	135499
CA (N; 11, 3^{12})	305797	268173	>day	267630	267816
CA (N; 12, 2^{14})	9422	8882	8972	8890	8905
CA (N; 13, 2^{14})	13251	11588	>day	10251	11059
CA (N; 14, 2^{15})	26579	23889	>day	23377	22651
CA (N; 15, 2^{16})	53977	45838	>day	46575	42839

جدول (۹): مقایسه از نظر کارایی در تولید یک نمونهٔ آزمون

	CPSO time	DPSO time	GS time	NEWPSO time
CA (2, 3^7)	0.28	2.14	0.41	0.21
CA (3, 3^7)	0.40	4.79	0.68	0.32
CA (4, 3^7)	0.82	9.86	0.66	0.65
CA (3, 3^8)	0.94	6.92	0.96	0.74
CA (3, 3^9)	1.35	9.82	1.21	0.91
CA (3, 3^{10})	1.82	13.72	2.02	1.21
CA (3, 4^7)	0.53	4.82	0.63	0.29
CA (3, 5^7)	0.57	7.96	0.57	0.42
CA (3, 6^7)	0.57	5.15	0.45	0.41
CA (3, 7^7)	0.58	3372	0.51	0.41

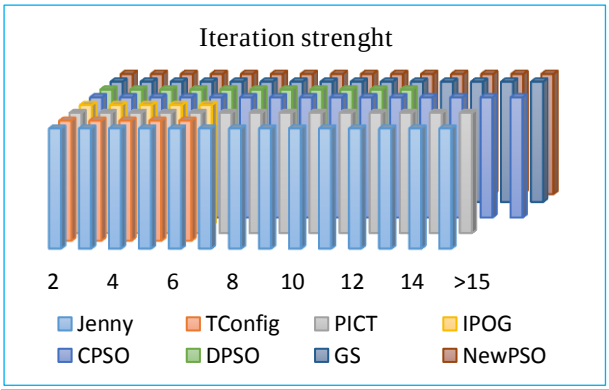
۳.۳. قوهٔ تعامل

یکی دیگر از معیارهای سنجش قدرت استراتژی در تولید CA، قوهٔ تعامل است. هرچه الگوریتم تعامل‌های بالا را پوشش دهد قوی‌تر است. در بین استراتژی در دسترس و مورد بررسی در

این مقاله، IPOG و TConfig تا t=6 توان تولید دنبالهٔ آزمون را دارند. سایر استراتژی قدرت پوشش تعاملات بالا را دارند. DPSO تا ۱۲، Jenny و PICT تا ۱۵ و سه استراتژی CPSO، GS و NEWPSO تعاملات بالای ۱۵ را نیز پشتیبانی می‌کنند شکل (۹) توان قوهٔ تعامل استراتژی‌ها را نشان می‌دهد.

جدول (۱۰): نتایج آزمون ویلکاکسون

		Ranks			Test Statistics
		NEW PSO<	NEW PSO>	NEW PSO=	Asymp. Sig. (2-tailed)
جدول ۵	NEWPSO-Jenny	12	0	1	.004
	NEWPSO-CPSO	7	0	6	.030
	NEWPSO-DPSO	5	1	7	.053
	NEWPSO-GS	0	0	13	.060
جدول ۹	NEWPSO-CPSO	10	0	0	.003
	NEWPSO-DPSO	10	0	0	.003
	NEWPSO-GS	10	0	0	.003



شکل (۹): مقایسهٔ استراتژی‌ها از نظر قوهٔ تعامل

۷. نتیجه‌گیری و کارهای آتی

استراتژی‌های مبتنی بر هوش مصنوعی به‌دلیل داشتن ساختار پیچیده و همچنین تکرارهای بسیار زیاد، نتایج بسیار خوبی را در تولید دنبالهٔ آزمون دارند. این ساختار پیچیده رابطهٔ معکوسی با زمان تولید و همچنین قوهٔ تعامل دارد؛ یعنی هرچه استراتژی پیچیدگی بالایی داشته باشد، تعامل کمتری را در زمان بیشتری تولید می‌کند. برای مثال می‌توان از الگوریتم‌های GA، SA و ACO نام برد که اغلب تا t=3 توان تولید دنبالهٔ آزمون را دارند و همچنین PSTG و CS توانستند با کاهش پیچدگی، تغییر ساختمان داده و افزایش سرعت تا t=6 را پشتیبانی کنند. DPSO نیز توانست تا t=12 را پوشش دهد و استراتژی HSS و CPSO تا t=15 را پشتیبانی می‌کنند.

دو استراتژی DPSO و CPSO توان پوشش تعاملات بالاتر را دارند اما برای تعاملات بالاتر به زمان بیش از یک روز نیاز دارند. DPSO از نظر efficiency بسیار قوی‌تر از سایر استراتژی‌های مبتنی بر هوش مصنوعی است، اما کارایی مناسبی ندارد. GS نیز از نظر زمان و قوهٔ تعامل قوی‌تر از استراتژی‌های مبتنی بر هوش مصنوعی است و تا t=20 را پوشش می‌دهد اما از نظر performance از DPSO ضعیف‌تر است. استراتژی‌های‌های مبتنی بر محاسبات نیز performance مناسبی دارند اما از نظر efficiency ضعیف‌تر از استراتژی‌های مبتنی بر هوش مصنوعی است. در حالت کلی، استراتژی که از نظر efficiency و performance قوی باشد و همچنین تعاملات بالا را پوشش دهد وجود ندارد. لذا ما در این پژوهش سعی کردیم با استفاده از الگوریتم PSO دنبالهٔ آزمونی با efficiency و performance مناسبی را تولید کنیم. در راهکار پیشنهادی ما از نقاط قوت دو استراتژی DPSO و GS بهره بردیم و توانستیم تا حدودی مزیت‌های هر دو را در قالب یک استراتژی که آن را NEWPSO نامیدیم، داشته باشیم. این استراتژی همچنین از یک تابع کمینه‌سازی ساده و مؤثر نیز استفاده می‌کند که efficiency را بهتر می‌کند. هرچند که استراتژی NEWPSO از نظر efficiency، performance و قوهٔ تعامل نتایج مناسبی دارد، به نظر می‌رسد این راه کماکان ادامه خواهد داشت. اصلی‌ترین مشکل تولید دنبالهٔ آزمون در CA گیر کردن در بهینهٔ محلی است. هرچند که DPSO یک راه‌حل مناسب ارائه داد، به نظر می‌رسد می‌توان در این زمینه تحقیقات بیشتری ارائه شود. همچنین برای افزایش سرعت استراتژی‌های مبتنی بر هوش مصنوعی می‌توان از روش OPAT نیز بهره گرفت یا از ترکیب الگوریتم‌های فرامکاشفه‌ای مانند [۲۹] استفاده کرد. همچنین می‌توان از ترکیب فازی و PSO [۳۰] یا از شبکهٔ عصبی [۳۱] در تولید آرایهٔ پوشش بهره گرفت.

مراجع

[1] Lin J., Luo C., Cai S., Su K., Hao D. and Zhang L., "*TCA: An Efficient Two-Mode Meta-Heuristic Algorithm for Combinatorial Test Generation (T)*," in 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), Lincoln, NE, USA, 9-13 Nov. 2015.

[2] Esfandyari S. and Rafe V., "*A tuned version of genetic algorithm for efficient test suite generation in interactive t-way testing strategy*," Information and Software Technology, vol. 94, pp. 165-185, 2018.

[3] Zamli K. Z., Din F., Kendall G. and Ahmed B. S., "*An Experimental Study of Hyper-Heuristic Selection and Acceptance Mechanism for Combinatorial t-way Test Suite Generation*," Information Sciences, vol. 399, pp. 121-153, 2017.

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

[5] Rafe V., "*Scenario-driven analysis of systems specified through graph transformations*," Journal of Visual Languages & Computing, vol. 24, pp. 136–145, 2013.

[6] Cohen M. B., Dwyer M. B. and Shi J., "*Interaction testing of highly-configurable systems in the presence of constraints*," in international symposium on Software testing and analysis, London, United Kingdom, 2007.

[7] Kennedy J. and Eberhart R., "*Particle swarm optimization*," in International Conference on Neural Networks, Perth, WA, Australia, Australia, 1995.

[8] Alsewari A. R. A. and Zamli K. Z., "*Design and implementation of a harmony-search-based variable-strengtht-way testing strategy with constraints support*," Information and Software Technology, vol. 54, no. 6, pp. 553–568, 2012.

[9] Cohen D. M., Dalal S. R., Fredman M. L. and Patton G. C., "*The AETG system: an approach to testing based on combinatorial design*," IEEE Transactions on Software Engineering, vol. 23, no. 7, pp. 437 - 444, 1997.

[10] Cohen M. B., "*Designing Test Suites for Software Interactions Testing*," PHD Thesis, University of Auckland, Department of Computer Science, Auckland, 2004.

[11] Czerwonka J., "*Pairwise testing in real world: practical extensions to test case generator*," in 24th Pacific Northwest Software Quality Conference, IEEE Computer Society, Portland, OR, USA, 2006.

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

[13] Bryce R. and Colbourn C. J., "*A density-based greedy algorithm for higher strength covering arrays*," Software Testing Verification and Reliability, vol. 17, no. 1, pp. 37-53, 2009.

[14] Lehmann E. and Wegener J., "*Test case design by means of the CTE XL*," in 8th European International Conference on Software Testing, Analysis & Review, Copenhagen, Denmark, 2000.

[15] Yu Y., Ng S. and Chan E., "*Generating, selecting and prioritizing test cases from specifications with tool support*," in Third International Conference on Quality Software, Dallas, TX, USA, USA, 2003.

[16] Arshem J., "*TVG download page*," 2019. [Online]. Available: http://sourceforge.net/projects/tvg.

[17] Tung Y.-W. and Aldiwan W., "*Automating test case generation for the new generation mission software system*," in IEEE Aerospace Conference. Proceedings (Cat. No.00TH8484), Big Sky, MT, USA, USA, 2000.

[18] Jenkins B., "*Jenny download web page*," Bob Jenkins' Website, 2019. [Online]. Available: http://burtleburtle.net/bob/math/jenny.html.

[19] Hartman A., "*IBM Intelligent Test Case Handler*," IBM alphaworks, 2019. [Online]. Available: http://www.alphaworks.ibm.com/tech/whitch.

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

[21] Calvagna A. and Gargantini A., "*IPO-s: incremental generation of combinatorial interaction test data based on symmetries of covering arrays*," in International Conference on Software Testing, Verification, and Validation Workshops, Denver, CO, USA, 2009.

[22] Lei Y., Kacker R., Kuhn D. R., Okun V. and Lawrence J., "*IPOG/IPOG-D: efficient test generation for multi-way combinatorial testing, Software Testing*," Software Testing, Verification & Reliability, vol. 18, no. 3, pp. 125-148, 2008.

[23] Kuhn R., "*ACTS page download*," 2019. [Online]. Available:http://csrc.nist.gov/groups/SNS/acts/download_tools.html.

[24] Stardom J., "*Metaheuristics and the Search for Covering and Packing Array*," Thesis (M.Sc.), Simon Fraser University, 2001, 2001.

[25] Shiba T., Tsuchiya T. and Kikuno T., "*Using artificial life techniques to generate test cases for combinatorial testing*," in 28th Annual International Computer Software and Applications Conference, Hong Kong, China, 2004.

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

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

[28] Wu H., Nie C., Kuo F.-C., Leung H. and Colbourn C. J., "*DPSO Page Download*," 2015. [Online]. Available: https://github.com/waynedd/DPSO.

[۲۹] سلیمی سرتختی، جواد، گلی بیدگلی، سلمان، «*ارائه یک الگوریتم ترکیبی با استفاده از الگوریتم کرم شب‌تاب، الگوریتم ژنتیک و جست‌وجوی محلی*»، مجله محاسبات نرم، جلد ۸، شماره ۱، ص ۱۴-۲۸، ۱۳۹۸.

[۳۰] قاسمی، رضا، محمدی، حمید رضا، طاهر، سید عباس «*کنترل فرکانس یک ریزشبکه جزیره‌ای با استفاده از کنترل هوشمند پاسخ گویی بار مبتنی بر منطق فازی و الگوریتم بهینه‌سازی ازدحام ذرات*»، مجله محاسبات نرم، جلد ۶، شماره ۲، ص ۱۸-۳۱، ۱۳۹۶.

[۳۱] مظفری شمسی، وجیهه، پیوندی، پدارم، «*پیش بینی خصوصیات نخ ریسیده شده در ریسندگی فاستونی با استفاده از روش ترکیبی شبکه عصبی با ناظر و بدون ناظر*»، مجله محاسبات نرم، جلد ۲، شماره ۲، ص ۶۲-۷۳، ۱۳۹۲.