Tabriz Journal of Electrical Engineering (TJEE), vol. 54, no. 3, Autumn 2024 Serial no. 109
DOI: 10.22034/tjee.2023.56934.4641

Optimal production of the test suite by the combinatorial testing method by applying changes in the gravitational search algorithm for the uniform strength cover array

Sajad Esfandyari[1], Leila Yousofvand[2], Einollah Pira[3], Vahid Rafe*[2]

Department of Computer Engineering, Faculty of Engineering, Malayer University, Malayer, Iran[1]
Department of Computer Engineering, Faculty of Engineering, Lorestan University, Khorramabad, Iran[2]
Faculty of Information Technology and Computer Engineering, Azarbaijan Shahid Madani University, Tabriz, 5375171379, Iran[3]
E-mails: esfandyari@malayeru.ac.ir;l.yousofvand@gmail.com; pira@azaruniv.ac.ir; v-rafe@araku.ac.ir;

Short Abstract
The need to increase the use of Combinatorial Testing (CT) in software testing has become a necessity in software development. CT is an efficient approach to reduce the size of the test suite so that the software can be tested with fewer test cases. Covering Array (CA) is one of the important branches in CT, which has different types .Many solutions have been provided for its production, which have appropriate efficiency (array size) and performance (speed). But there is a lack of a solution that has both efficiency and performance. In this research, we have tried to produce an optimized test suite(with the minimum number of test cases) by using the gravitational search algorithm(GSA) and changing the neighbor selection method. Also, by changing the structure of the data and giving weight to the parameters not covered, we have increased the speed of producing the test suite. The weighting of non-covered parameters and the change in the behavior of the gravity algorithm have caused a smart search to find non-covered test cases. This increase in speed has made the proposed solution capable of producing test suites for high-power configurations. Also, the evaluation results show that the proposed solution outperforms other popular algorithms such as the genetic algorithm, the particle mass search algorithm, and even the gravity search algorithm itself.

Keywords
Software Testing, Combinatorial Testing (CT), Covering Array (CA), Gravitational Search Algorithm (GSA).

1- Short Introduction

Combinatorial Testing (CT) is a common test in software testing to detect errors caused by interactions of subsystems of a system. The Covering Array (CA) is one of the branches of the CT. This method can test the system by sampling test cases and a small number of test cases. These test cases should be such that they satisfy the test objectives.. Also, the number of test items should be minimal and at an appropriate speed. The solutions for producing the CA are divided into two general categories: mathematical methods and computational methods. Mathematical methods generally use well-known algebraic and combinatorial methods, and computational methods use greedy and meta-heuristic methods to produce the CA. Computational methods are more discussed due to having more suitable results in terms of productivity, and in recent years, the use of computational methods has become more common. In this research, by using the Gravitational Search Algorithm (GSA) and optimizing this algorithm, we will produce the minimum test suite with appropriate efficiency and performance.

2- Proposed Work and Methodology

In this research, we have tried to produce the minimum test suite with appropriate efficiency by using the GSA and optimizing this algorithm. We have tried to increase the speed and accuracy of the test suite generation by changing the data structure. One of the main problems in the generation of minimal test sets using meta-heuristic algorithms is falling into local optima. This also creates problems for the gravity search algorithm. In this solution, by generating random test cases in each step, we have been able to acceptably overcome the mentioned problem. And also, according to the information we get from the proposed data structure, we make the search method smarter in the GSA, which has a considerable impact on efficiency and performance. The proposed solution is implemented in Java. We performed our experiments on a computer with Windows 7 OS, 6GB RAM, 2.20GHz core ™ i7QM CPU, and jdk 1.8. In the evaluation section, we have compared our solution with several available. It can be seen that the proposed solution has acceptable results in terms of efficiency compared to other related solutions.

3- Conclusion

uniform strength covering array is one of the important branches of CT The main problem in CA is that meta-heuristic algorithms usually reduce the fitness of individuals after applying changes to individuals. The GSA algorithm is no exception to this rule and the velocity of applying individuals fitness decreases. In this research, we applied changes in the GSA algorithm that can generat a test suite with a smaller number of test cases. Also, in this solution, to increase the speed, changes have been applied in the data structure, which can calculate the weight of the test cases and the total production time. To reduce the size of test suite, one of the best solutions for generating CA is the DPSO algorithm. The results in the evaluation section show that the proposed solution is much stronger than this solution.

4- References

H. Wu, C. Nie, F.-C. 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.

S. Esfandyari and V. Rafe, "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.

I. Izquierdo-Marquez, J. Torres-Jimenez, B. Acevedo-Juárez and H. Avila-George, "A greedy-metaheuristic 3-stage approach to construct covering arrays," *Information Sciences*, vol. 460, pp. 172-189, 2016.

تولید بهینه مجموعه آزمون به روش آزمون ترکیبی با اعمال تغییر در الگوریتم جستجوی گرانشی برای آرایه پوشش با قوه ثابت

سجاد اسفندیاری
استادیار گروه مهندسی کامپیوتر، دانشکده فنی و مهندسی، دانشگاه ملایر، ملایر، ایران

لیلا یوسفوند
گروه مهندسی کامپیوتر، دانشکده فنی مهندسی، دانشگاه لرستان، خرم آباد، ایران

وحید رافع
دانشیار، دانشکده فنی مهندسی، دانشگاه اراک، ایران

عین الله پیرا
استادیار، دانشکده فناوری اطلاعات و مهندسی کامپیوتر، دانشگاه شهید مدنی آذربایجان، تبریز، ایران

چکیده
لزوم افزایش استفاده از آزمون ترکیبی در نرم‌افزارهای امروزی به امری ضروری در توسعه نرم‌افزار تبدیل شده است. آزمون ترکیبی یا آزمون t-way راهکاری کارآمد در کاهش حجم مجموعه آزمون است به نحوی که می‌توان نرم‌افزار را با نمونه آزمون‌های کمتری مورد آزمون قرار داد. پوشش آرایه یکی از شاخه‌های مهم در آزمون ترکیبی است که انواع مختلفی دارد . راهکارهای فراوانی برای تولید آن ارائه شده است که کارایی (سرعت) و بهره‌وری (اندازه آرایه) مناسبی را دارند. اما خلاء راهکارای که هر دو خصوصیت کارایی و بهره‌وری را توامان داشته باشد به چشم می‌خورد.
در این پژوهش سعی شده است با استفاده از الگوریتم جستجوی گرانشی و تغییر در نحوه انتخاب همسایه‌ها، موفق به تولید مجموعه آزمون بهینه شده‌ایم و هچنین با تغییر در ساختمان داده‌ها و وزن دهی به پارامترهای پوشش داده نشده سرعت تولید مجموعه آزمون را افزایش داده‌ایم. وزن دهی به پارامترهای پوشش داده نشده و تغییر در رفتار الگورتیم گرانشی سبب جستجوی هوشمندانه جهت یافتن نمونه آزمون‌های پوشش داده نشده، شده است. افزایش سرعت باعث شده است که راهکار پیشنهادی توان تولید مجموعه آزمون برای پیکربندی‌های بزرگ را داشته باشد و همچنین نتایج آزمون نشان می‌دهد که راهکار پیشنهادی از الگوریتم‌های مطرح مانند الگوریتم ژنتیک، الگوریتم جستجوی توده ذرات و حتی خود الگوریتم جستجوی گرانشی نتایج بسیار بهتری را دارد.

کلمات کلیدی
آزمون ترکیبی، آزمون نرم‌افزار، الگوریتم جستجوی گرانشی

نام نویسنده مسئول: وحید رافع
ایمیل نویسنده مسئول: v-rafe@araku.ac.ir

تاریخ ارسال مقاله: ۱۴۰۲/۰۳/۱۲
تاریخ(های) اصلاح مقاله: ۱۴۰۲/۰۸/۱۲
تاریخ پذیرش مقاله: ۱۴۰۲/۱۰/۰۳

۱- مقدمه

آزمون ترکیبی[1] یک تست رایج در آزمون نرم‌افزار برای تشخیص خطاهای ناشی از تعامل‌های زیرسیستم‌های یک سیستم است [1]. آرایه پوشش[2] یکی از شاخه‌های آزمون ترکیبی است این آزمون با نمونه برداری تعداد کمی از نمونه آزمون‌ها می‌تواند سیستم را مورد آزمون قرار دهد [2]. این نمونه آزمون‌ها باید به گونه‌ای باشد که اهداف آزمون را برآورده کنند و همچنین تعداد نمونه آزمون‌ها حداقل و با سرعت مناسب باشد [3].

کاربرد آزمون ترکیبی برای سیستم‌های بزرگ بیشتر به چشم می‌خورد برای مثال سیستمی با ۹ پارامتر که هر پارامتر آن ۵ مقدار را اختیار می‌کند را در نظر بگیرید؛ برای آزمون این سیستم نیاز به بررسی 5^9 (۱،۹۵۳،۱۲۵) نمونه آزمون است که رقم بسیار بزرگی است لذا کاهش این تعداد نمونه آزمون امری حیاطی است [4].

آرایه پوشش با قوه متغیر[3] و آرایه پوشش با محدودیت[4] نیز از خانواده آزمون ترکیبی هستند. راهکارهای تولید آرایه پوشش به دو دسته کلی تقسیم می‌شوند: روش‌های ریاضی[5] و روش‌های محاسباتی[6] [5]. روش‌های ریاضی عموما از روش-

[1] Combinatorial Testing (CT)
[2] Covering Array (CA)
[3] Variable Strength Covering Array (VCA)
[4] Constraints Covering Array (CCA)
[5] Mathematical Methods
[6] Computational Methods

های شناخته شده جبری و ترکیبی استفاده می‌کنند و روش‌های محاسباتی از روش‌های حریصانه و فراابتکاری برای تولیدآرایه پوشش استفاده می‌کنند [6].

روش‌های محاسباتی به‌دلیل داشتن نتایج مناسب‌تر از نظر بهره‌وری، بیشتر مورد بحث قرار می‌گیرند و در سال‌های اخیر نیز استفاده از روش‌های محاسباتی عمومیت بیشتری داشته است [7]. راهکارهای بسیار متنوعی با استفاده از الگوریتم‌های هوش مصنوعی ارایه شده است که برخی از آن‌ها عبارت‌اند از: شبیه سازی تبرید[7] [8, 9]، الگوریتم تپه نوردی [10]، الگوریتم ژنتیک[8] [2]، الگوریتم‌های چند فازی [11, 12, 13, 14, 15] الگوریتم کلونی مورچگان[9] [16] و الگوریتم توده ذرات[10] [1, 17, 18]. راهکارهای نام برده از نظر بهره‌وری[11] نتایج بسیار مطلوبی را دارند که در این بین راهکار DPSO[12] [1] بهترین عملکرد را دارد اما از نظرکارایی[13] (زمان) مناسب نیست و GS[14] [2] از نظر کارایی بسیار مناسب است اما بهره‌وری مناسبی را در قیاس با DPSO ندارد لذا مطالعه برای پیدایش روشی که کارایی و بهره‌وری را همزمان داشته باشد، لازم است.

در این پژوهش سعی شده است با بهره‌گیری از الگوریتم جستجوی گرانشی و بهینه سازی این الگوریتم مجموعه آزمون کمینه با کارایی مناسب رو تولید کنیم. برای تولید مجموعه آزمون با سرعت بالا سعی شده است با تغییر در ساختمان داده استفاده شده در [2] سرعت و دقت تولید مجموعه آزمون را افزایش دهیم همچنین در این راهکار با استفاده ار تابع کمینه‌ساز استفاده شده در [3] تعدادی از نمونه آزمون‌ها از مجموعه آزمون کم می‌کند و باعث بهینه شدن مجموعه آزمون می‌شود. یکی از مشکلات مهم در تولید مجموعه آزمون کمینه با استفاده از الگوریتم‌های فرامکاشفه‌ای، گیرکردن در بهینه محلی است که به نحوی گریبان‌گیر الگوریتم جستجوی گرانشی نیز است که ما در این راهکار در هر گام با تولید یک نمونه آزمون تصادفی توانسته‌ایم تا حد قابل قبولی بر مشکل یاد شده غلبه کنیم و همچنین با به اطلاعاتی که از ساختمان داده‌ها بدست می‌آوریم نحوه جستجو در الگوریتم گرانشی را هوشمند می‌کنیم که این امر در سرعت و دقت تاثیر بسزایی را ایفا می‌کند.

برای نشان دادن برتری راهکار پیشنهادی در بخش ارزیابی راهکار خود را با سه راهکار شناخته شده مبتنی بر محاسبات (Jenny, TConfig و PICT) و چهار راهکار برتر مبتنی بر هوش محاسباتی که عبارتنداز DPSO، GS، APSO و همچنین پیاده سازی نسخه خام GSA را مقایسه کرده‌ایم که نتایج ارزیابی را با استفاده از آزمون آماری ویلکاکسون مورد نقد و بررسی قرار داده‌ایم و در این ارزیابی مشخص است که راهکار پیشنهادی نتایج بسیار مطلوبی را نسبت به رقبای خود تولید می‌کند.

سهم و نوآوری راهکار پیشنهادی عبارتنداز:

- ✓ تغییر در ساختار ساختمان داده جهت دسترسی سریع و آسان به نمونه آزمون‌های پوشش داده نشده جهت افزایش کارایی.
- ✓ وزن دهی به پارامترهای پوشش داده نشده در ساختمان داده‌ها جهت پیدا کردن نمونه آزمون با وزن بیشتر و افزایش بهره‌وری.
- ✓ تغییر روش انتخاب kbest در الگوریتم GSA و کمک به بهبود کارایی و بهره‌وری.

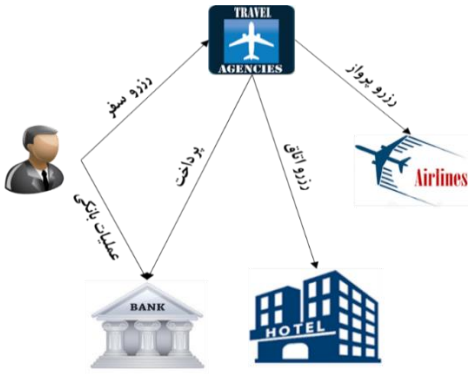
ادامه مقاله به این شرح است: فصل اول با عنوان مفاهیم بنیادی به تشریح آرایه پوشش و الگوریتم جستجوی گرانشی می‌پردازد. در فصل دوم جزئیات پیاده‌سازی الگوریتم جستجوی گرانشی برای تولید آرایه پوشش با قوه ثابت شرح داده می‌شود. فصل ۴ مربوط به ارزیابی راهکار پیشنهادی است و بخش فصل ۵ به نتیجه‌گیری و پیشنهاد برای کارهای آتی در نظر گرفته‌شده است.

۲- مفاهیم بنیادی

در این بخش سعی خواهیم کرد توضیحات لازم را در خصوص آرایه پوشش و الگوریتم GSA را به‌طور کامل شرح دهیم و همچنین در انتهای این بخش راهکارهای پیشین در حوزه آرایه پوشش به‌طور کامل مورد نقد و بررسی قرار خواهند گرفت.

۲-۱- آرایه پوشش

اگر فرض کنیم که سیستم تحت آزمون با n پارامتر کنترل می‌شود که این پارامترها شامل متغیرهای ورودی، رفتار سیستم و نظایر آن است. i امین پارامتر v_i مقدار دارد در نتیجه تاپل $(x_1, x_2, \dots x_n)$ یک نمونه آزمون برای سیستم تحت آزمون است که $x_i \in v_i$ و $1 \le i \le n$ است. برای تشریح آرایه پوشش از مثال مطرح شده در [19] استفاده می‌کنیم. این سیستم شامل ۵ پارامتر است که عبارت‌اند از: هتل، آژانس مسافرتی، کاربر، خط هوایی و بانک (شکل ۱) کامپوننت دیاگرام مثال آژانس مسافرتی را نشان می‌دهد. پارامترهای سیستم مستقل از یکدیگر هستند و هر مقداری را می‌توانند اتخاذ کنند. برای تشخیص خطا یا بعبارتی آزمون کامل سیستم آژانس مسافرتی می‌بایست تمام حالت‌های سیستم مورد آزمون قرارگیرد که تعداد آن ۲*۲*۳*۲*۳=۷۲ نمونه آزمون است اما آزمون ترکیبی تنها بخشی از ۷۲ نمونه آزمون را برای آزمون سیستم نیاز دارد. فرض کنید در سیستم آژانس مسافرتی کلانیت (۱) مجاز به عملیات بانکی از بانک ملت نیست لذا نمونه آزمون‌های (ملت, -,-,-, اسفندیاری) موجب رخ دادن خطا می‌شوند لذا مجموعه آزمونی که به روش آزمون ترکیبی، سیستم را آزمون می‌کند حداقل یک بار باید دوتایی (ملت, -,-,-, اسفندیاری) را پوشش بدهد.



شکل ۱- ساختار یک آژانس مسافرتی

یکی از رایج‌ترین مجموعه آزمونی که در آزمون ترکیبی مورد استفاده قرار می‌گیرد آرایه پوشش است. در واقع آرایه پوشش (یا t-way testing که t قوه تعامل آزمون است.) یک آرایه N × p که تعداد نمونه آزمون های آن N است و دارای دو ویژگی زیر است [1]:

- ✓ هر ستون i شامل عضوهایی از مجموعه V_i باشد.
- ✓ تعداد سطرهای هر زیر آرایه N × t از آن، تمام ترکیبات $|V_{p1}| \times |V_{p2}| \times \dots \times |V_{pt}|$ از t ستون را حداقل یکبار پوشش دهد.

آرایه پوشش را با نماد $CA(N; t, p, x_1, x_2, \dots, x_p)$ و در صورتی که $x_1 = x_2 = \dots = x_p = x$ با نماد $CA\ (N;\ t, p, v)$ و یا $CA\ (N;\ t, v^p)$

[7] Simulated Annealing (SA
[8] genetic algorithm (GA)
[9] ant colony optimization (ACO)
[10] Particle Swarm Optimization (PSO)

[11] efficiency
[12] Discrete Particle Swarm Optimization
[13] perfomance
[14] Genetic Strategy

نمایش داده می‌شود. در اکثر منابع اگر x_i ها با هم برای نباشند آن را MCA15 و اگر t متغیر آن را آرایه پوشش با قوه متغیر می‌نامند (VSCA16) [2].

جدول ۱- گزینه‌های آژانس مسافرتی

مشتری	آژانس مسافرتی	خط هوایی	هتل	بانک
رافع	رخش	زاگرس	پارسیان	سپه
اسفندیاری	پاسارگاد	کاسپین	گلستان	پاسارگاد
-	-	آسمان	-	ملت

فرض کنید در سیستم آژانس مسافرتی کاربرد (۱) مجاز به عملیات بانکی از بانک ملت نیست لذا نمونه آزمون‌های (ملت, -,-,-, اسفندیاری) موجب رخ دادن خطا می‌شوند بنابراین مجموعه آزمونی که به روش آزمون ترکیبی، سیستم را آزمون می‌کند حداقل یک بار باید دوتایی (ملت, -,-,-, اسفندیاری) را پوشش بدهد. همانطور که در جدول ۲ مشاهده می‌شود برای آزمون آژانس مسافرتی با آزمون ترکیبی بجای ۷۲ نمونه آزمون فقط به ۹ نمونه آزمون نیاز است.

برای درک بیشتر از آزمون ترکیبی به سیستمی با ۴ پارامتر که هر کدام ۲ مقدار (۰ و ۱) را به خود اختصاص می‌دهند، می‌پردازیم. این سیستم دارای ۲۴ = (v^p) نمونه آزمون است (جدول ۲) یعنی برای آزمون می‌بایست هر ۱۶ نمونه آزمون برای آزمون به سیستم داده شود. حال اگر بخواهیم این سیستم را با پیکربندی CA با قوه تعامل ۲ مورد آزمون قرار دهیم تعداد نمونه آزمون‌های آن به ۵ کاهش می‌یابد (شکل۲). پوشش در CA به‌گونه‌ای است که نمونه آزمون‌ها باید تمامیه ترکیبات سیستم را پوشش دهند در این سیستم شش ترکیب وجود دارد (AB, AC, AD, BC, BD, CD) که در بین نمونه آزمون باید برای هر ترکیب چهار مقدار ۰۰، ۰۱، ۱۰ و ۱۱ وجود داشته باشد و تا زمانی که تمامی حالات پوشش داده شود. تعداد کل پوشش‌ها زمانی که تمامی پارامترها مقدار یکسانی دارند از رابطه (۱) و در غیر این صورت از رابطه (۲) بدست می‌آید.

۲-۲- الگوریتم جستجوی گرانشی

در الگوریتم جستجوی گرانشی[17] [20] ، بهینه‌یابی به کمک طرح قوانین گرانشی و حرکت در یک سیستم مصنوعی با زمان گسسته انجام می‌شود. محیط سیستم همان محدوده تعریف مسأله است. طبق قانون گرانش، هر جرم، محل و وضعیت سایر اجرام را از طریق قانون جاذبه گرانشی درک می‌کند. بنابراین میتوان از این نیرو به عنوان ابزاری برای تبادل اطلاعات استفاده کرد. از بهینه-یاب طراحی شده برای حل هر مسئله بهینه‌سازی که در آن هر جواب مسئله به صورت یک موقعیت در فضا قابل تعریف و میزان شباهت آن با سایر جواب‌های مسئله به صورت یک فاصله قابل بیان باشد، می‌توان استفاده کرد. میزان اجرام با توجه به تابع هدف تعیین می‌شوند. به دلیل جلوگیری از افزایش حجم مقاله از ارائه جزییات این الگوریتم خودداری می‌کنیم.

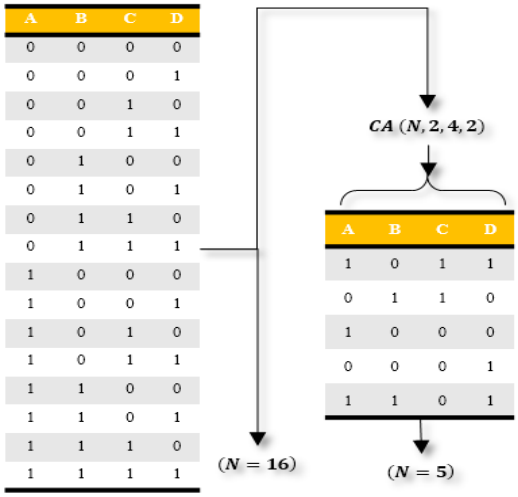
[15] Mixed Coverage Array
[16] Variable Strength Covering Array
[17] Gravitational Search Algorithm (GSA).

جدول۲-مجموعه آزمون برای پیکربندی CA $(9; 2, 2^3, 3^2)$

مشتری	آژانس مسافرتی	خط هوایی	هتل	بانک
رافع	رخش	کاسپین	پارسیان	ملت
اسفندیاری	پاسارگاد	زاگرس	گلستان	ملت
رافع	پاسارگاد	آسمان	پارسیان	سپه
اسفندیاری	رخش	آسمان	گلستان	پاسارگاد
رافع	رخش	زاگرس	گلستان	سپه
اسفندیاری	پاسارگاد	کاسپین	پارسیان	پاسارگاد
رافع	رخش	زاگرس	پارسیان	پاسارگاد
اسفندیاری	پاسارگاد	کاسپین	گلستان	سپه
اسفندیاری	رخش	آسمان	گلستان	ملت

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

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



شکل ۲- ساختار آرایه پوشش

۲-۳- پژوهش‌های پیشین

در این بخش ما به بررسی دو دسته از تحقیقات پیشین می‌پردازیم. دسته اول شامل بررسی راهکارهای تولید CA اعم از روش‌های ریاضی و محاسباتی است و دسته دوم شامل راهکارهای استخراج پارامترهای سیستم تحت آزمون برای CA می‌باشد.

روش‌های ریاضی[18] و روش‌های محاسباتی[19] دو روش اصلی در تولید آزمون ترکیبی است. روش‌های ریاضی قالبا از آرایه متعامد[20] برای تولید مجموعه آزمون استفاده می‌کنند که بررسی این روش‌ها خارج از بحث این مقاله است اما عمده تحقیقات پیشین بر روی روش‌های محاسباتی صورت گرفته است که خود به دو دسته مبتنی بر محاسبات محض و مبتنی بر هوش مصنوعی تقسیم می-شوند [5].

در حالت کلیدو روش کلی برای تولید مجموعه آزمون وجود دارد، در روش اول در هر زمان یک نمونه آزمون به مجموعه آزمون اضافه می‌شود تا زمانی که مجموعه آزمون تمام حالت ها را پوشش دهد که این روش را One Test At

[18] Mathematical methods
[19] Computational methods
[20] Orthogonal arrays

Time(OTAT) می‌نامند در واقع در این روش گسترش به صورت افقی صورت می‌گیرد اما روش دیگر که One Parameter At Time (OPAT) نام دارد ابتدا پوشش کامل با دو پارامتر تکمیل می‌شود سپس پارامتر سوم به مجموعه آزمون اضافه می‌شود (گسترش عمودی) و با اضافه شدن نمونه آزمون سه پارامتری مجموعه آزمون تکمیل می‌شود (گسترش افقی) در واقع در این روش گسترش به صورت افقی و عمودی صورت می‌پذیرد. در حالت کلی خانواده IPOG از روش OPAT و سایر روش‌ها از OTAT استفاده می‌کنند که در ادامه به بررسی برخی از آن‌ها پرداخته می‌شود.

راهکار AETG [21] که به دو صورت mAETG و mAETG توسعه داده شد اولین روشی بود که با OATA مجموعه آزمون تولید کرد در این روش در هر تکرار چند نمونه آزمون را انتخاب می‌کند و سپس در بین آن‌ها نمونه آزمونی که تعداد تعاملات بیشتری را پوشش داده است به مجموعه آزمون اضافه می‌کند و این روال را تا زمانی که پوشش کامل شود، ادامه می‌یابد در واقع می‌توان این روش را یک روش حریصانه نام برد.

راهکار PICT [22] دیگر راهکار مبتنی بر OTAT است که به صورت تصادفی از بین نمونه آزمون‌های تولیدی یکی را انتخاب می‌کند انتخاب نمونه آزمون به صورت تصادفی باعث شده است که نتواند نتایج مطلوبی از نظر بهره‌وری را نسبت به سایر راهکارها تولید کند از نقاط قوت این راهکار می‌توان به کارایی مطلوب و همچنین تولید مجموعه آزمون تا قوه ۱۰ اشاره کرد همچنین این راهکار توانایی پشتیبانی از VSCA را دارد.

راهکار Jenny [23] دیگر راهکار شناخته شده در در OTAT است این راهکار نتایجی مشابه با PICT تولید می‌کند.

راهکار GTWay [24] یکی از قدرتمندترین راهکارهای محاسباتی است این راهکار نیر مبتنی بر OTAT است نقطه قوت این راهکار ذخیره سازی نمونه آزمون‌های پوشش داده نشده بصورت ساختار بیتی است که همین امر باعث شده است که این راهکار تا قوه تعامل ۱۲ مجموعه آزمون تولید کند این راهکار از نظر زمان و بهره‌وری نتایج بسیار خوبی را دارد و می‌توان گفت در بین راهکار-های محاسباتی از نظر زمان بهترین نتایج را تولید می‌کند.

همانطورکه اشاره شد دسته دوم راهکارهای مبتنی بر محاسبات، الگوریتم-های هوش مصنوعی هستند که این راهکارها نیز قالبا مبتنی بر OATA هستند تعداد الگوریتم‌های فرامکاشفه‌ای که تاکنون استفاده‌شده‌اند بسیار زیاد است که برخی از آن‌ها عبارتند از: SA [8, 9]، TS [25, 26]، GA [2]، PSO [18, 17]، CS [27]، HSS [5]، [28] GSA و بسیاری دیگر که در مقاله [29] مورد ارزیابی قرار گرفته‌اند.

ابتدا در سال ۲۰۰۱ استاردم [28] با سه الگوریتم SA، GA و TS مجموعه آزمون را با قوه تعامل ۲ تولید کرد که SA نتایج بهتری را داشت پس از آن کوهن در [29]، SA را تقویت کرد به نحوی که توانست قوه تعامل ۳ را نیز پشتیانی کند این راهکار به دلیل استتفاده از ساختارپیچیده و همچنین استفاده از الگوریتم فشرده سازی از زمان مناسبی برخوردار نمی‌باشد.

در ادامه احمد و همکاران در [17] راهکار مبتنی بر الگوریتم PSO ارائه دادند که آن‌را PSTG نامید و توان پشتیبانی پیکربندی‌های تا قوه ۶ را دارد. همچنین این راهکار توان پشتیبانی از VSCA را دارد. همچنین احمد و همکاران در [27] با استفاده از الگوریتم فاخته مجموعه آزمون را تا قوه ۶ تولید کردند که نتایج نسبتا بهتری را از نظر زمان و بهره‌وری نسبت به PSTG دارد اما توان تولید VSCA را ندارد. دیگر راهکار قدرتمند در حوزه تست ترکیبی، راهکار HSS است که مبتنی بر الگوریتم جستجوی هارمونی است این راهکار توان تولید تا قوه ۱۵ را دارد و از نظر بهره‌وری نیز نتایجی مشابهی با راهکارهای PSTG و CS را دارد. درحالت کلی الگوریتم‌های فرامکاشفه‌ای نتایج مناسبی از نظر بهره-وری برای تولید مجموعه آزمون ندارد این مشکل در [1] برای اولین بار مطرح شد و با استفاده از الگوریتم PSO راهکاری را ارائه داد که توانست نتایجی بسیار مناسبی را از نظر بهره‌وری تولید کند اما از نظر زمان خیر. در این راهکار با ارائه مثالی نشان داد که اعمال سرعت[21] لزوماً کیفیت فرد را بالا نمی‌برد و حتی ممکن است نتیجه عکس داشته باشد که راهکار ارائه شده روشی مناسب برای غلبه بر این مشکل است. همچنین این راهکار می‌تواند قوهای تعامل بالای ۶ و VSCA را نیز پشتیبانی کند.

مشکل اصلی در تولید آرایه پوشش با الگوریتم‌های مبتنی بر توده ذرات این است که اعمال سرعت لزوما نتیجه را بهتر نخواهد کرد و حتی در برخی از موارد نتیجه 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 بهره وری را داشت اما از نظر کارایی مناسب نبود. ساختار پیاده سازی 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 [2]. دیگر استراتژی که در [1] ارایه شد CPSO بود که این استراتژی از نظر کارایی از DPSO قوی تر بود اما بهره وری DPSO قدرت بیشتری دارد.

از دیگر راهکارهای که توان پشتیبانی CA و VSCA را دارد GS [2] است که مبتنی برا الگوریتم ژنتیک می‌باشد این راهکار با ایجاد تغییراتی در ساختمان داده سرعت تولید را بالا برده و همچنین نتایج مناسب را از نظر بهره‌وری دارد و پیکربندی‌های با قوه تعامل ۲۰ را نیز پشتیبانی می‌کند. در این راهکار در هر سطر حالات پوشش داده نشده مربوط به یک ترکیب ذخیره می‌شود و برای مشخص شدن تفاوت بین ترکیب‌ها، خود ترکیب نیز در همان سطر ذخیره می‌شود سپس با تبدیل مقدار نمونه آزمون به دیسیمال به راحتی به سلول مورد نظر دسترسی پیدا می‌کند و پوشش یا عدم پوشش نمونه آزمون را بررسی می‌کند.

دیگر راهکاری که در این مقاله مورد بحث قرار می‌گیرد الگوریتم GALP است که در سال ۲۰۲۱ [32] ارائه شده است. این راهکار از ترکیب دو الگوریتم GA و PSOSLV [30] برای تولید مجموعه آزمون بهره گرفته است و توانسته است نتایج مناسبی را از نظر زمانی و بهره‌وری تولید بدست آورد.

۳- راهکار پیشنهادی

برای تولید مجموعه آزمون کمینه در این پژوهش از الگوریتم جستجوی گرانشی استفاده می‌شود که در ادامه به تشریح گام‌های پیاده‌سازی آن پرداخته می‌شود.

۳-۱- گام ۱: ایجاد جمعیت اولیه

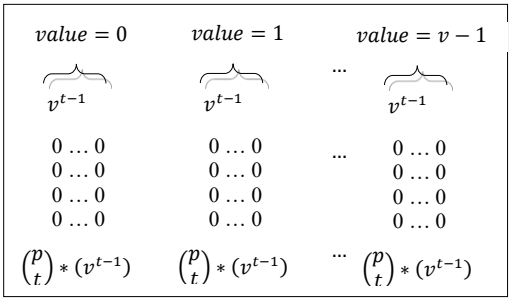
در این مرحله مقدار دهی اولیه پارامترهای مربوط به الگوریتم GSA تعیین می‌شود و همچنین جمعیت اولیه نیز به صورت تصادفی ایجاد می‌گردد. هر جرم در الگوریتم جستجوی گرانشی معادل یک نمونه آزمون است. این جرم به تعداد پارامترهای یک پیکربندی دارای سلول است که هر کدام از این سلول‌ها مقداری بین 1 تا $v-1$ را اختیار می‌کنند.

۳-۲- گام ۲: ایجاد ماتریس پوشش

محاسبه وزن یک نمونه آزمون یکی از مهم‌ترین گام‌های پیاده‌سازی الگوریتم است برای محاسبه وزن ابتدا ماتریسی که آن را ماتریس پوشش می-نامند [۲] ایجاد می‌کنیم. تعداد سطرهای این ماتریس C_t^p است؛ و هر سطر دارای دو بخش است بخش اول شماره ترکیب را ذخیره می‌کند و بخش دوم

[21] velocity

دارای $|v_t| * \dots * |v_2| * |v_1|$ سلول است (رابطه (۱) و (۲))که در حالت پیش-فرض مقدار ۰ (به معنای عدم پوشش) را دارد. بعد از هر پوشش سلول متناظر به یک تبدیل می‌شود زمانی که تمام صفرها به یک تبدیل شود پوشش کامل می‌گردد و الگوریتم خاتمه می‌یابد. در این راهکار برای دسترسی سریع‌تر به مقادیر پوشش داده نشده ستون‌ها را به v دسته تقسیم می‌کنیم.

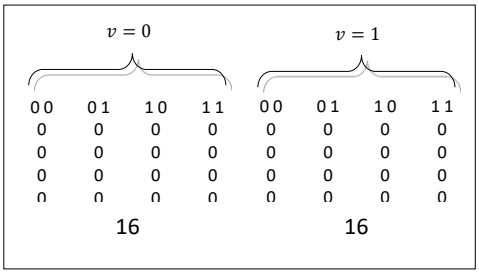


شکل ۳- ساختمان داده پیشنهادی

برای مثال در پیکربندی CA (N; 3, 4^2)، از آنجایی که v = 2 است طبق رابطه (۱) تعداد کل نمونه آزمون‌ها ۳۲ بیت است در راهکار پیشنهادی ستون‌ها به دو بخش تقسیم می‌شوند که هر بخش به ۱۶ بیت تقسیم می‌شوند (شکل ۴) این نحوه ذخیره سازی باعث می‌شود که در هر مرحله ما نصف ماتریس پوشش را جستجو نکنیم که در سرعت محاسبه وزن یک نمونه آزمون بسیار تاثیر گذار است.

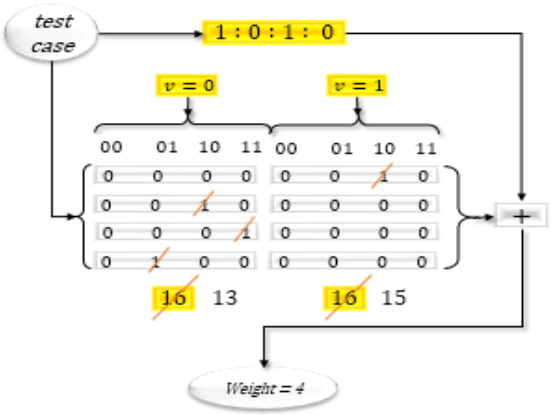
۳-۳- گام ۳: محاسبه وزن یک نمونه آزمون

نحوه محاسبه وزن یک نمونه آزمون (۱۰۱۰) در شکل ۵ نشان داده شده است در این پیکربندی نیاز به بررسی چهار ترکیب (۱،۲،۳)، (۱،۲،۴)، (۱،۳،۴) و (۲،۳،۴) که به ترتیب نمایانگر مقادیر (۰۱۰)، (۱۱۰)، (۱۰۰) و (۱۰۱) هستند که سه مقدار اول در بخش اول (v = 0) و مقدار چهارم نیز در بخش دوم (v = 1) جستجو می‌شود و در صورت عدم پوشش مقادیر ۰ (عدم پوشش) در ماتریس به ۱ (پوشش داده شده) تبدیل می‌شوند.



شکل ۴- مقداردهی اولیه ساختمان داده پیشنهادی

در این مثال از آنجایی که تمامی مقادیر در حالت عدم پوشش قرار دارند لذا وزن این نمونه آزمون ۴ بوده و از مقدار ۳۲ کسر خواهد شد درنتیجه تعداد حالات پوشش داده نشده به ۲۸ بروز می‌شود و این روال تا زمانی که این مقدار به صفر برسد ادامه داده می‌شود. همانطور که در شکل ۵ ملاحظه می‌شود تعداد حالت های پوشش داده نشده در مرحله به ترتیب ۱۳ و ۱۵ نمونه است ما از این مقادیر در الگوریتم جستجوی گرانشی برای سریع‌تر رسیدن به پوشش کامل استفاده می‌کنیم به‌نحوی که مقادیر جدید را به سمت عدد ۱۵ سوق می‌دهیم.



شکل ۵- نحوه محاسبه وزن یک نمونه آزمون

۴-۳- گام ۴: تولید آرایه پوشش با الگوریتم جستجوی گرانشی

در این قسمت سعی می‌کنیم با اعمال تغییراتی در الگوریتم جستجوی گرانشی، مجموعه آزمون بهینه‌ای را تولید کنیم. برای تولید مجموعه آزمون با الگوریتم GSA هر نمونه آزمون معادل یک جسم است که شامل p (تعداد پارامترها) جزء است و هر جز مقداری بین ۰ و $v-1$ را به خود اختصاص می‌دهد و وزن هر جسم نیز با توجه به شکل ۵ محاسبه می‌شود و جسم با بیشترین وزن را $best$ می‌نامیم. یکی از عوامل مهم در الگوریتم GSA انتخاب $Kbest$ است که ما در این پژوهش به‌جای استفاده از استخراج $Kbest$ به روش معمولی، k نمونه جدید را از $best$ ایجاد می‌کنیم در این روش با توجه به تعداد جمعیت مقدار k را بین ۲ تا ۱۰ در نظر می‌گیریم سپس به‌طور تصادفی تعدادی از مقادیر آن جایگزین می‌گردد. برای اثبات عملکرد مناسب راهکار پیشنهادی، فرض می‌کنیم که تعداد ۷ نمونه آزمون به مجموعه آزمون اضافه شده است (شکل ۶ (الف)). که ماتریس پوشش آن بعد از به‌روزرسانی به صورت شکل ۶ (ب) می‌شود اعداد صفر در شکل ۶ (ب) به معنی عدم پوشش است.

تغییر مهم‌تر در الگوریتم GSA در مرحله اعمال سرعت بر جسم است. انتظار می‌رود در این گام با اعمال سرعت، وزن جسم بیشتر یا بدون تغییر بماند اما همانطور که در [1] برای الگوریتم PSO مشاهده شد که با اعمال سرعت، وزن جسم کاهش می‌ابد در الگوریتم GSA نیز وضعیت مشابهی به وجود می‌آید و بالغ بر ۷۰ درصد، وزن جسم کم می‌شود. برای مثال مجموعه آزمون شکل ۶ را درنظر بگیرید. این مجموعه آزمون برای پیکربندی CA (N; 3, 6^3) در حال تکمیل شدن است یکی از اجسام (نمونه آزمون) با مقدار 1101 (p0p1p2p3) دارای وزن 1 است که پس اعمال سرعت به مقدار 0001 تغییر خواهد کرد که این نمونه آزمون جدید وزنی برابر ۰ را دارد (شکل ۷).

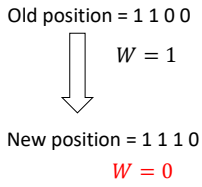
(الف)

مجموعه آزمون	
1	0001
2	0111
3	0100
4	1000
5	1110
6	0010
7	1011
8	-------

(ب)

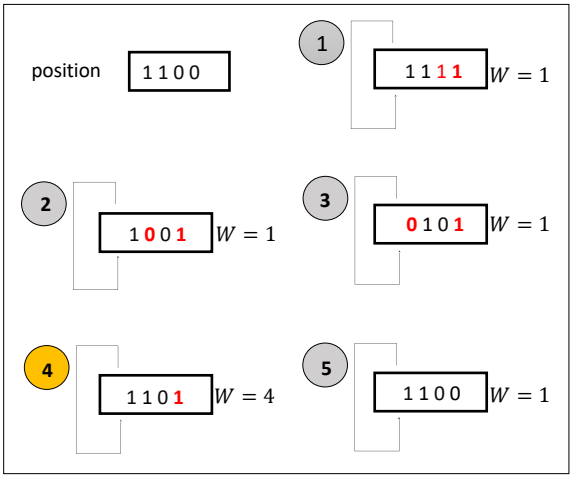
v = 0	v = 1
123 1110	1111
124 1111	1110
134 1111	1011
234 1111	1011
1	3

شکل ۶- مثال ساختمان داده پیشنهادی



شکل ۷- وضعیت وزن یک نمونه آزمون پس از اعمال سرعت

یکی از دلایل مشکل فوق می‌تواند تغییر غیر هوشمندانه سلول‌های نمونه آزمون باشد. در الگوریتم‌های فرامکاشفه‌ای قالبا با یک تغییر کوچک نتایج بهتری را مشاهده می‌کنیم اما در تولید آزمون ترکیبی با استفاده از الگوریتم‌های فرامکاشفه‌ای معمولا نتایج مناسبی را مشاهده نمی‌کنیم. راهکارهای ارایه شده است که در اصل الگوریتم‌ها تغییراتی اعمال می‌کنند تا نتایج بهتری مشاهده شود مهمترین و قوی‌ترین این تغییر در [1] ملاحظه می‌شود. ما در این قسمت قصد داریم راهکاری را برای الگوریتم GSA ارایه دهیم که تغییرات سلول‌های نمونه آزمون به صورت هوشمندانه به سمت نمونه آزمون‌های پوشش داده نشده پیش برود. به منظور رسیدن به این هدف، در هر سطر از ساختمان داده‌ی پیشنهادی تعدادِ نمونه آزمون‌های پوشش داده نشده ذخیره می‌شود. هر نمونه آزمونی که در سطر مربوطه پوشش داده می‌شود یک واحد از مقدار آخرین سلول در هر سطر کسر می‌شود بنابراین هرچه این مقدار بیشتر باشد به معنای این است که سلول‌های نمونه آزمون در این ترکیب نیاز به تغییر بیشتری دارند تا احتمال یافتن ترکیب دلخواه بیشتر شود. همچنین در هرگروه نیز تعداد نمونه آزمون‌های پوشش داده نشده را ذخیره می‌کنیم همانطورکه در شکل ملاحظه می‌شود در قسمت $v = 0$ تعداد نمونه آزمون های پوشش داده نشده برابر ۱ و در $v = 1$ این تعداد برابر ۳ است که این مقادیر تاثیر مستقیم بر سلول‌های ۳ و ۴ از نمونه آزمون را دارند. لذا می‌توان نتیجه گرفت که مقادیر این سلول‌ها می‌بایست به سمت ۱ میل کند (۷۵٪ به سمت ۱ و ۲۵٪ به سمت ۰). مقادیر ستون آخر ماتریس پوشش نیز تاثیر بر دو سلول ۱ و ۲ از نمونه آزمون را دارند که در این مثال در صد احتمال برابر است (۵۰٪ به سمت ۰ و ۵۰٪ به سمت ۱). حال با این احتمالات ۵ نمونه آزمون جدید از *position* ایجاد می‌شود (شکل ۸) و وزن هریک محاسبه و در صورت بهتر بودن از best با آن جایگزین می‌گردد. همچنین مقدار G به G (G1G2G3G4) = [0. 5, 0. 5, 0.25, 0.75] بروز می‌شود. مراحل اجرای الگوریتم پیشنهادی در شکل ۹ نشان داده شده است.

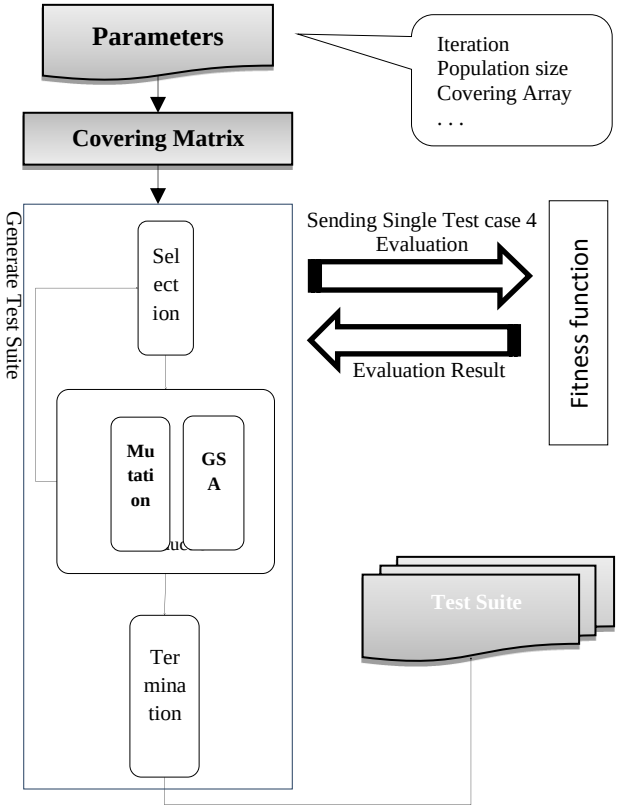


شکل ۸- انتخاب هوشمندانه kbest

۴- ارزیابی

در این بخش قصد داریم نتایج راهکار پیشنهادی را مورد ارزیابی قرار دهیم

در ابتدا راهکار پیشنهادی با الگوریتم GSA و یکی از بهترین الگوریتم‌های در دسترس در تولید CA یعنی DPSO (که راهکاری برای غلبه بر مشکل گیرکردن در بهینه محلی را ارائه داده است) و همچنین الگوریتم GSA با روال عادی پیاده‌سازی شده است که روش پیشنهادی را با این الگوریتم‌ها مقایسه می‌کنیم (جداول۵-۷) همانطور که می‌دانیم الگوریتم‌های ابتکاری قطعی نیستند و ممکن است در اجراهای مختلف نتایج مختلفی را نیز مشاهده کنیم لذا ما برای ارزیابی بهتر هر پیکربندی را صد بار اجرا کرده‌ایم و درنهایت میانگین و بهترین نتایج را در جداول ۵-۷ نشان می‌دهیم. بهترین مقدار در هر سطر به صورت ضخیم نشان داده می‌شود. همچنین در این ارزیابی ما سعی کرده‌ایم از لحاظ زمان تولید نیز راهکارها مورد ارزیابی قرار گیرند. برای ارزیابی زمانی ما الگوریتم‌ها را در شرایط یکسان و بر روی یک سخت افزار (Windows 7, RAM 6GB، CPU 2.20GHz core™ i7QM و jdk 1.8) اجرا کرده‌ایم که در ادامه به شرح آن می‌پردازیم همچنین جدول ۴ مقادیر پارامترهای الگوریتم‌ها را نشان می‌دهد.



شکل ۹- مراحل تولید مجموعه آزمون در راهکار پیشنهادی

جدول ۴- پارامتر های الگوریتم پیشنهادی

الگوریتم	پارامتر	مقدار
GSA	Population	۸۰-۱۰
	Elitist Check	۱
	a	۲۰
	G0	۱۰۰

اولین ارزیابی در این مقاله مربوط به ۱۱ پیکربندی برای t = 2 است که نتایج ارزیابی در جدول ۵ نشان داده شده است. راهکارها در این ارزیابی به دو دسته محاسبات محض و مبتنی بر هوش مصنوعی تقسیم می‌شوند راهکارهای محاسبات محض شامل Jenny، TConfig و PICT هستند و راهکار های مبتنی بر هوش مصنوعی نیز شامل CS، PICT، DPSO، GA، GSA و و GSA است.

همانطور که در جدول ۵ مشاهده می‌شود راهکارهای مبتنی بر هوش مصنوعی بسیار قوی‌تر از راهکارهای محاسباتی هستند که در این بین راهکار پیشنهادی و DPSO نتایج بهتری را دارند.

نتایج ارزیابی برای پیکربندی‌های با قوه (strength) ۳ ($t = 3$) در جدول ۶ آورده شده است در این جدول نیز قدرت الگوریتم‌های فرامکاشفه‌ای نمایان است. در بین الگوریتم‌های الگوریتم های فرامکاشفه‌ای نیز سه الگوریتم DPSO و GA و GSA از سایر الگوریتم‌ها قوی‌تر هستند.

ارزیابی بعدی مربوط به $t > 3$ است (جدول ۷). در این ارزیابی نیز راهکار-های فرامکاشفه‌ای بسیار قوی‌تر از راهکارهای مبتنی برمحاسبات هستند در این ارزیابی علاوه بر بهره‌وری قدرت تعامل نیز نمایان هست همانطور که ملاحظه می‌شود راهکار TConfig، CS و PSTG تا قوه ۶ قادر به تولید مجموعه آزمون هستند. الگوریتم DPSO برای برخی از پیکربندی‌های بزرگ به زمان بیش از یک روز نیاز ارد و سایر الگوریتم‌ها قادر هستند تمامی پیکربندی‌های موجود در جدول ۷ را پشتیبانی کند. در این ارزیابی راهکار GSA از سایر راهکارها قوی‌تر است.

جدول ۵- مقایسه راهکار پیشنهادی با راهکارهای پیشین در $t = 2$

	راهکارهای محاسبات محض			راهکارهای مبتنی بر الگوریتم‌های فراابتکار					
	Jenny N	TConfig N	PICT N	PSTG N.Best	DPSO N.Best	GA N.Best	GSA [28] N.Best	**GSA** N.Best	**GSA** N.AVG
CA (N; 2, 2^7)	۸	۷	۷	**۶**	۷	**۶**	**۶**	**۶**	**۶.۳**
CA (N; 2, 3^3)	**۹**	۱۰	۱۰	**۹**	**۹**	**۹**	**۹**	**۹**	**۹.۸**
CA (N; 2, 3^4)	۱۳	۱۰	۱۳	**۹**	**۹**	**۹**	**۹**	**۹**	**۹.۸**
CA (N; 2, 3^5)	۱۴	۱۴	۱۳	۱۲	**۱۱**	**۱۱**	۱۲	**۱۱**	**۱۱.۹**
CA (N; 2, 3^6)	۱۵	۱۵	۱۴	**۱۳**	۱۴	**۱۳**	**۱۳**	**۱۳**	۱۴.۱
CA (N; 2, 3^7)	۱۶	۱۵	۱۶	۱۵	**۱۴**	**۱۴**	۱۵	**۱۴**	**۱۴.۶**
CA (N; 2, 3^8)	۱۷	۱۷	۱۶	**۱۵**	**۱۵**	**۱۵**	**۱۵**	**۱۵**	۱۶.۲
CA (N; 2, 3^9)	۱۸	۱۷	۱۷	۱۷	**۱۵**	**۱۵**	۱۷	**۱۵**	۱۶.۱
CA (N; 2, 3^{10})	۱۹	۱۷	۱۸	۱۷	**۱۶**	**۱۶**	۱۷	**۱۶**	۱۷.۳
CA (N; 2, 4^7)	۲۸	۲۸	۲۷	۲۶	۲۴	۲۴	۲۸	**۲۳**	۲۵.۷
CA (N; 2, 5^7)	۳۷	۴۰	۴۰	۳۷	**۳۴**	۳۶	۳۷	**۳۴**	۳۷.۸

جدول ۶- مقایسه راهکار پیشنهادی با راهکارهای پیشین در $t = 3$

	راهکارهای محاسبات محض			راهکارهای مبتنی بر الگوریتم‌های فراابتکار					
	Jenny N	TConfig N	PICT N	PSTG N.Best	DPSO N.Best	GA N.Best	GSA[28] N.Best	**GSA** N.Best	**GSA** N.AVG
CA (N; 3, 2^7)	۱۴	۱۶	۱۵	۱۳	۱۵	**۱۲**	۱۳	**۱۲**	**۱۲.۶**
CA (N; 3, 3^4)	۳۴	۳۲	۳۴	**۲۷**	**۲۷**	**۲۷**	**۲۷**	**۲۷**	**۲۷.۸**
CA (N; 3, 3^5)	۴۰	۴۰	۴۳	۳۹	۴۱	۳۷	۳۹	**۳۶**	۳۹.۹
CA (N; 3, 2^{10})	۱۸	۲۰	۱۸	۱۷	**۱۶**	**۱۶**	۱۷	**۱۶**	**۱۶.۸**
CA (N; 3, 3^6)	۵۱	۴۸	۴۸	۴۵	**۳۳**	۳۷	۴۵	۳۶	۳۸.۶
CA (N; 3, 3^7)	۵۱	۵۵	۵۱	۵۰	**۴۸**	**۴۸**	۵۰	**۴۸**	**۴۸.۹**
CA (N; 3, 3^8)	۵۸	۵۸	۵۹	۵۴	**۵۲**	**۵۲**	۵۴	**۵۲**	۵۴.۸
CA (N; 3, 3^9)	۶۲	۶۴	۶۳	۵۸	**۵۶**	**۵۶**	۵۸	**۵۶**	۵۷.۷
CA (N; 3, 3^{10})	۶۵	۶۸	۶۵	۶۲	**۵۹**	**۵۹**	۶۲	**۵۹**	۶۱.۹
CA (N; 3, 3^{11})	۶۵	۷۲	۷۰	۶۴	**۶۳**	**۶۳**	۶۴	**۶۳**	۶۵.۴

جدول ۷- مقایسه راهکار پیشنهادی با راهکارهای پیشین در $t > 3$

	راهکارهای محاسبات محض			راهکارهای مبتنی بر الگوریتم‌های فراابتکار					
	Jenny N	TConfig N	PICT N	PSTG N.Best	DPSO N.Best	GA N.Best	GSA[28] N.Best	**GSA** N.Best	**GSA** N.AVG
CA (N; 4, 2^7)	۳۱	۳۶	۳۲	۲۹	۳۳۴	**۲۷**	**۲۷**	**۲۷**	**۲۷.۹**
CA (N; 4, 2^{10})	۳۹	۴۵	۴۳	۳۴	۳۴	**۲۵**	۳۴	۳۴	۳۴.۵
CA (N; 5, 3^7)	۴۵۸	۴۷۷	۴۵۲	۴۴۱	۴۳۸	**۴۳۱**	۴۳۸	**۴۳۱**	۴۳۸.۲
CA (N; 6, 3^8)	۱۴۶۶	۱۵۱۵	۱۴۵۵	۱۴۰۱	۱۴۰۹	۱۳۹۵	۱۴۰۰	**۱۳۸۹**	۱۴۰۲.۲
CA (N; 7, 3^9)	۴۷۴۶	>day	۴۶۱۸	NS	۴۴۵۱	۴۴۳۷	۴۴۵۱	**۴۴۲۸**	۴۴۵۳.۷
CA (N; 8, 3^{10})	۱۴۹۹۹	>day	۱۴۵۹۹	NS	۱۳۹۹	**۱۳۹۰۷**	۱۳۹۵۲	۱۳۹۱۴	۱۳۹۴۲.۲
CA (N; 9, 3^{11})	۴۷۰۰۹	>day	۴۵۵۲۱	NS	>day	۴۳۸۰۸	۴۳۸۱۵	**۴۳۴۴۰**	۴۳۴۴.۵
CA (N; 10, 3^{12})	۱۴۷۰۰۴	>day	۱۴۱۹۹۰۱	NS	>day	۱۳۶۰۹۶	۱۳۵۸۶۹	**۱۳۵۲۴۵**	۱۳۶۱۰۹.۵
CA (N; 11, 3^{12})	۳۰۵۷۹۷	>day	۲۷۸۹۹۳	NS	>day	۲۶۷۶۳۰	۲۶۷۳۶۸	**۲۶۷۶۲۰**	۲۶۷۷۶۸۴.۵
CA (N; 12, 2^{14})	۹۴۲۲	>day	۹۱۱۲	NS	۸۹۷۲	۸۸۹۰	۸۸۹۳	**۸۸۸۸**	۸۹۱۴.۲
CA (N; 13, 2^{14})	۱۳۲۵۱	>day	۱۲۴۴۱	NS	>day	**۱۰۲۵۱**	۱۱۲۱۰	۱۰۶۴۶	۱۰۶۸۲.۵
CA (N; 14, 2^{15})	۲۶۵۷۹	>day	۲۵۰۶۶	NS	>day	۲۳۳۷۷	۲۳۳۱۲	**۲۰۷۲۱**	۲۳۳۸۷.۲
CA (N; 15, 2^{16})	۵۳۹۷۷	>day	۵۱۱۲۷	NS	>day	۴۶۵۷۵	۴۵۵۷۵	**۴۲۳۶۳**	۴۶۵۹۷.۸

۵- نتیجه گیری و آینده پیش‌رو

آزمون ترکیبی یکی از زیر شاخه‌های آزمون نرم افزار است که با استفاده از آزمون t-way اقدام به تولید مجموعه آزمون می‌کنید. مجموعه آزمون تولید شده ویژگی‌های متفاوتی دارد که مهم‌ترین آن‌ها آرایه پوشش است. در راستای تولید آرایه پوشش راهکارهای بسیار متعدی ارایه شده است که هدف اصلی آن‌ها کاش تعداد نمونه آزمون‌های مجموعه آزمون هستند. در این راستا با دو دسته عمده راهکارهای محاسبات محض (که Jenny، TConfig و PICT از مهم‌ترین آن‌ها بحساب می‌آیند) و الگوریتم‌های مبتنی بر هوش مصنوعی شامل PSO، GA، GSA و نظایر آن. یکی از اصلی‌ترین مشکلات تولید آرایه پوشش برای الگوریتم‌های هوش مصنوعی گیرکردن در بهینه محلی است که راهکارهای متعددی برا رفع این مشکل ارایه شده استکه اصلی‌ترین آن‌ها DPSO است. برای غلبه بر این مشکل با مطالعاتی که بر روی الگویتم‌های فراابتکاری داشته‌ایم در ادامه پژوهش‌های خود سعی کردیم با اعمال تغییرات اساسی در ساختمان داده‌های مربوط به ذخیره‌سازی نمونه آزمون‌های پوشش داده نشده و تغییر در رفتار الگوریتم GSA موفق به تولید مجموعه آزمون کمینه شده‌ایم برای اثبات ادعای خود راهکار پیشنهادی را با تعدادی از الگوریتم‌های شناخته شده مقایسه نموده و در این مقایسه ملاحظه می‌شود راهکار پیشنهادی نتایج مطلوبی را در مقایسه با رقبای خود تولید می‌کند.

برای مقایسه آماری راهکار پیشنهادی با سایر راهکارها از روش آزمون Wilcoxon signed rank sum استفاده می‌کنیم. این ابزار که در SPSS قرار دارد. برای داده‌های ورودی بین دو راهکار دو خروجی دارد: Test Statistics و ranks که برای دو راهکار A و B، ranks سه مقدار دارد: تعداد نمونه های A > B، A < B و A = B. و Test Statistics نیز دو خروجی دارد: z و Asymp. Sig. (2-tailed). اهمیت z در این گفتار به چشم نمی‌خورد؛ مقدار Asymp. Sig. (2-tailed) عددی بین ۰ و ۱ است که به زبان ساده اگر این مقدار کمتر از ۰.۰۵ باشد به معنای آن است که بین A و B اختلاف معنایی وجود دارد. نتایج آزمون Wilcoxon در جدول ۸، جدول ۹ و جدول ۱۰ نشان داده شده است. که همانطور که ملاحظه می‌شود راهکار پیشنهادی از لحاظ آماری برتری خود را نسبت به راهکارهای هوش مصنوعی و محاسبات محض نشان می‌دهد.

جدول ۸- نتایج آزمون ویلکاکسون برای جدول ۵

	Ranks			Test Statistics	
	GSA <	GSA >	GS A =	Z	Asymp. Sig. (2-tailed)
Jenny– GSA	۱۰	۰	۱	-۲.۸۴	۰.۰۰۴۵
TConfig–GSA	۱۱	۰	۰	-۲.۹۶	۰.۰۰۲۹
PICT– GSA	۱۱	۰	۰	-۲.۹۶	۰.۰۰۳۰
PSTG– GSA	۶	۰	۵	-۲.۲۳	۰.۰۲۵
DPSO– GSA	۳	۰	۸	-۱.۷۳	۰.۰۸۳
GA– GSA	۲	۰	۹	-۱.۳۴۱	۰.۱۷۹
GSA– GSA	۶	۰	۵	-۲.۲۲۵	۰.۰۲۶۰

الگوریتم‌های مبتنی بر هوش مصنوعی قالبا از روش OTAT برای تولید نمونه آزمون استفاده می‌کنند این روش‌ها از دقت بسیار بالایی برخوردار هستند اما نسبت به برخی از الگوریتم‌های مبتنی بر OPAT سرعت پایین‌تری را دارند از جمله این الگوریتم‌ها می‌توان IPOG را نام برد که برای قوه‌های بسیار بالا سرعت مناسبی را دارند به عنوان یه راهکار نوین در آینده می‌توان ترکیب راهکارهای مبتنی بر هوش مصنوعی و IPOG را پیشنهاد داد. از دیگر راهکارهای پیشنهادی برای کارهای آتی می‌توان به استفاده کاربردی آرایه پوشش در حوزه‌های مختلف مانند شبکه‌های عصبی کانولوشنال [31, 32]، سرورهای ابری [33]، محلی‌سازی اشکال[22] [34]، سیستم‌های فازی-عصبی [35, 36, 37] انتخاب ویژگی [37, 38] و نظایر آن را نام برد. همچنین با توجه به افزایش روزافزون الگوریتم‌های فراابتکاری می‌توان این الگوریتم‌های نو ظهور را در جهت بهبود کارایی و بهره‌وری آرایه پوشش بکار بست.

[22] bug localization

جدول ۹- نتایج آزمون ویلکاکسون برای جدول ۶

	Ranks			Test Statistics	
	GSA <	GSA >	GSA =	Z	Asymp. Sig. (2-tailed)
Jenny– GSA	۱۰	۰	۰	-۲.۸۴۰	۰.۰۰۴۸
TConfig– GSA	۱۰	۰	۰	-۲.۸۴۰	۰.۰۰۴۸
PICT– GSA	۱۰	۰	۰	-۲.۸۴۰	۰.۰۰۴۸
PSTG– GSA	۹	۰	۱	-۲.۶۸۶	۰.۰۰۷۲
DPSO– GSA	۲	۱	۷	-۰.۸۱۶	۰.۴۱۴۲
GA– GSA	۲	۰	۸	-۱.۴۱۴	۰.۱۵۷۲
GSA– GSA	۹	۰	۱	-۲.۶۸۶	۰.۰۰۷۲

جدول ۱۰- نتایج آزمون ویلکاکسون برای جدول ۷

	Ranks			Test Statistics	
	GSA <	GSA >	GSA =	Z	Asymp. Sig. (2-tailed)
Jenny– GSA	۱۳	۰	۰	-۳.۱۷	۰.۰۰۱۴
TConfig–GSA	۴	۰	۰	-۱.۸۲	۰.۰۶۷۸
PICT– GSA	۱۳	۰	۰	-۳.۱۷	۰.۰۰۱۴
PSTG– GSA	۶	۰	۱	-۲.۲۰	۰.۰۲۷۲
DPSO– GSA	۸	۳	۲	-۱.۳۷	۰.۱۶۷۹
GA– GSA	۱۰	۱	۲	-۲.۴۰	۰.۱۶۳۶

مراجع

[1] H. Wu, C. Nie, F.-C. 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.

[2] S. Esfandyari and V. Rafe, "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] S. Esfandyari and V. Rafe, "GALP: a hybrid artificial intelligence algorithm for generating covering array," *soft computing*, vol. 25, p. 7673–7689, 2021.

[4] S. Esfandyari and V. Rafe, "Extracting Combinatorial Test parameters and their values using model checking and evolutionary algorithms," *Applied Soft Computing*, vol. 91, pp. 1-19, 2020.

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

[6] E. Pira, V. Rafe and S. Esfandyari, "Minimum Covering Array Generation Using Success-History and Linear Population Size Reduction based Adaptive Differential Evolution Algorithm," *TABRIZ JOURNAL OF ELECTRICAL ENGINEERING*, vol. 52, no. 2, pp. 77-89, 2022.

[7] Z. Abbasi, S. Esfandyari and V. Rafe, "Covering array generation using teaching learning base optimization algorithm," *Tabriz Journal of Electrical Engineering*, vol. 48, no. 1, pp. 161-171, 2018.

[8] J. Torres-Jimenez and E. Rodriguez-Tello, "Simulated annealing for constructing binary covering arrays of variable strength," in *in Proc. Congr.Evol. Comput*, Barcelona, Spain, Jul., 2010.

[9] J. T.-J. a. E. Rodriguez-Tello, "New bounds for binary covering arrays using simulated annealing," *Inf. Sci.*, vol. 185, no. 1, pp. 137-152, 2012.

[10] S. Esfandyari and V. Rafe, "A Hybrid solution for Software testing to minimum test suite generation using hill climbing and bat search algorithms," *Tabriz Journal of Electrical Engineering*, vol. 46, no. 3, pp. 25-35, 2016.

[11] E. Pira, V. Rafe and S. Esfandyari, "A three-phase approach to improve the functionality of t-way strategy," *Soft Computing*, pp. 1-21, 2023.

[12] M. Farokhian, J. Shoaei and S. Esfandyari, "GWC: A tool for automatic web data extraction," in *13th Symposium on Advances in Science and Technology: Sustainable Land of Computer and Information Technology*, 2018.

[13] I. Izquierdo-Marquez, J. Torres-Jimenez, B. Acevedo-Juárez and H. Avila-George, "A greedy-metaheuristic 3-stage approach to construct covering arrays," *Information Sciences*, vol. 460, pp. 172-189, 2016.

[14] J. Torres-Jimenez, D. O. Ramirez-Acuna, B. Acevedo-Juárez and H. Avila-George, "New upper bounds for sequence Covering Arrays using a 3-stage approach," *Expert Systems with Applications*, vol. 207, p. 118022, 2022.

[15] J. Torres-Jimenez, H. Avila-George and I. Izquierdo-Marquez, "A two-stage algorithm for combinatorial testing," *Optimization Letters*, vol. 11, no. 3, pp. 457-469, 2017.

[16] V. Rafe, M. Darghayedi and E. Pira, "MS-ACO: a multi-stage ant colony optimization to refute complex software systems specified through graph transformation," *Soft Computing*, pp. 1-26, 2018.

[17] 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, p. 1330–1347, 2012.

[18] S. Esfandyari and V. Rafe, "Using the Particle Swarm Optimization Algorithm to Generate the Minimum Test Suite in Covering Array with Uniform Strength," *Soft Computing Journal*, vol. 8, no. 2, pp. 66-79, 2021.

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

[20] E. Rashedi, H. Nezamabadi-pour and S. Saryazdi, "GSA: a gravitational search algorithm," *Information Sciences, 179(13)*, vol. 179, no. 13, pp. 2232-2248, 2009.

[21] 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.

[22] 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, 2006.

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

[24] K. Z. Zamli, M. F. J. Klaib, M. I. Younis, N. A. M. Isa and R. Abdullah, "Design and implementation of a t-way test data generation strategy with automated execution tool support," *Information Sciences*, vol. 181, no. 9, pp. 1741-1758, 2011.

[25] K. Z. Zamli, B. Y. Alkazemi and G. Kendall, "A Tabu Search hyper-heuristic strategy for t-way test suite generation," vol. 44, pp. 57-74, 2016.

[26] L. Gonzalez-Hernandez, N. Rangel-Valdez and J. Torres-Jimenez, "Construction of mixed covering arrays of variable strength using a tabu search approach," in *International Conference on Combinatorial Optimization and Applications*, Berlin, Heidelberg, 2010.

[27] B. S. Ahmed, T. Sh. 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, p. 13–29, 2015.

[28] K. M. Htay, R. R. Othman, A. Amir and J. M. H. Alkanaani, "Gravitational search algorithm based strategy for combinatorial t-way test suite generation," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 8, pp. 4860-4873, 2022.

[29] A. A. Muazu, A. S. Hashim and A. Sarlan, "Review of Nature Inspired Metaheuristic Algorithm Selection for Combinatorial t-Way Testing," *IEEE Access*, vol. 10, pp. 27404 - 27431, 2022.

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

[31] M. B. Cohen, C. J. Colbourn, and A. C. Ling, "Constructing strength three covering arrays with augmented annealing," *Discrete Math.*, vol. 308, no. 13, p. 2709–2722, 2008.

[32] S. Esfandyari and V. Rafe, "Correction to: GALP: a hybrid artificial intelligence algorithm for generating covering array," *Soft Computing*, 2021.

[33] E. S. A. Shahri, A. Alfi and J. Machado, "Fractional fixed-structure H∞ controller design using Augmented Lagrangian Particle Swarm Optimization with Fractional Order Velocity," *Applied Soft Computing*, vol. 77, pp. 688-695, 2019.

[34] D. Giveki, "Improving the performance of convolutional neural networks for image classification," *Optical Memory and Neural Networks*, vol. 30, pp. 51-66, 2021.

[35] H. Rastegar and D. Giveki, "Designing a new deep convolutional neural network for content-based image retrieval with relevance feedback," *Computers and Electrical Engineering*, vol. 106, p. 108593, 2023.

[36] A. Nasri, "Energy-Efficient Cloud Servers: an overview of Solutions and Architectures," *Journal of Computer & Robotics*, vol. 13, no. 1, pp. 33-44, 2020.

[37] L. Y. S. S. V. Rafe, "Automatic bug localization using a combination of deep learning and model transformation through node classification," *Software Quality Journal*, pp. 1-19, 2023.

[38] H. Eftekhari, "Neuro-fuzzy cooperative collision warning system in based on driver behavior in chain accident using connected vehicles," *Journal of Transportation Research*, vol. 20, no. 2, pp. 339-352, 2023.

[39] H. R. Eftekhari and M. Ghatee, "A similarity-based neuro-fuzzy modeling for driving behavior recognition applying fusion of smartphone sensors," *Journal of Intelligent Transportation Systems*, vol. 23, no. 1, pp. 72-83, 2019.

[40] R. B. Zadeh, M. Ghatee and H. R. Eftekhari, "Three-phases smartphone-based warning system to protect vulnerable road users under fuzzy conditions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 7, pp. 2086-2098, 2017.

[41] V. Nosrati and M. Rahmani, "Diversity improvement in homogeneous ensemble feature selection: a case study of its impact on classification performance," *Neural Computing and Applications*, 2023.

[42] A. Rafiee, P. Moradi and A. Ghaderzadeh, "A swarm intelligence based multi-label feature selection method hybridized with a local search strategy," *Tabriz Journal of Electrical Engineering*, vol. 51, no. 4, pp. 443-454, 2022.