راهکاری نوین جهت تولید دنباله آزمون کمینه در فرآیند آزمون نرم‌افزار با ترکیب الگوریتم‌های جستجوی تپه‌نوردی و جستجوی خفاش

سجاد اسفندیاری[۱]، دانشجوی کارشناسی ارشد، وحید رافع[۲]، دانشیار

۱- دانشکده فنی مهندسی - دانشگاه اراک - اراک - ایران - s-esfandyari@arshad.araku.ac.ir
۲- دانشکده فنی مهندسی - دانشگاه اراک - اراک - ایران - v-rafe@araku.ac.ir

چکیده: امروزه استفاده از الگوریتم‌های مبتنی بر هوش جمعی به همراه استراتژی آزمون T-ستونی[۱] در حوزه تولید خودکار دنباله آزمون کمینه، افزایش یافته است. در این میان الگوریتم‌های جستجوی ژنتیک، الگوریتم مورچگان، شبیه‌سازی تبرید، ازدحام توده ذرات و الگوریتم جستجوی ممنوعه سهم به سزایی را دارند. اکثر این الگوریتم‌ها به دلیل دا شتن ساختار پیچیده و ا ستفاده از محا سبات د شوار قادر به تولید دنباله آزمون برای مقدار T>۳ نمی‌باشند. در این پژوهش با ترکیب الگوریتم جستجوی تپه‌نوردی و الگوریتم جستجوی خفاش، دنباله آزمون بهینه با استفاده از استراتژی آزمون T-ستونی برای پیکربندی‌های مختلف تولید می‌شود. این راهکار قادر است که دنباله آزمون تا مقدار T=۱۰ را نیز تولید کند. یکی از معیارهای ارزیابی تولید دنباله آزمون، اندازه آرایه تولیدشده است. در این پژوهش ضمن مقایسه راهکار پیشنهادی با جدیدترین الگوریتم‌های منتشرشده در حوزه تولید خودکار دنباله آزمون، برتری آن نیز نشان داده خواهد شد.

واژه‌های کلیدی: آزمون نرم‌افزار، الگوریتم خفاش، الگوریتم تپه‌نوردی، تولید نمونه آزمون.

A Hybrid solution for Software testing to minimum test suite generation using hill climbing and bat search algorithms

S. Esfandyari, M.Sc Student[1], V. Rafe, Associate Professor[2]

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

Abstract: Nowadays using meta-heuristic algorithms besides T-way testing strategy is increasing to generate minimum test suites automatically. Genetic Algorithm, Ant Colony, Simulated Annealing, and Tabu Search play an important role in this regard. However, most of these algorithms cannot generate test suits efficiently for T>3 due to their complex structure and complicated computations. In this paper, we propose a hybrid approach using hill climbing and bat search algorithms to minimum test suit generation. Our proposed solution uses T-way strategy to test suit generation for different configuration of the system. The proposed solution can generate test suits up to T=10. Since one of the most important criterions for the evaluation of test suits is the array size, hence we compare our results with other existing approaches in terms of this criterion. Our results show that our proposed solution outperforms other approaches.

Keywords: Software testing, bat algorithm, hill climbing algorithm, test case generation

تاریخ ارسال مقاله: ۹۴/۰۶/۰۷
تاریخ اصلاح مقاله: ۹۴/۰۷/۲۸
تاریخ پذیرش مقاله: ۱۳۹۴/۱۰/۲۰
نام نویسنده مسئول: وحید رافع
نشانی نویسنده مسئول: ایران - اراک - سردشت - پردیس - دانشگاه اراک - دانشکده فنی و مهندسی - گروه کامپیوتر

۱- مقدمه

هدف از آزمون نرم‌افزار، یافتن خطاهای موجود در نیازمندی‌ها، طراحی و پیاده‌سازی آن است که این کار باعث افزایش اعتماد به کیفیت نرم افزار می شود [۲-۱]. فرآیند آزمون، شامل تولید داده‌های آزمون، دادن این داده‌ها به‌عنوان ورودی به نرم‌افزار تحت‌بررسی و سپس بررسی نتایج حاصل از آزمون نرم‌افزار است. این فرآیند را می‌توان به شکل دستی انجام داد ولی این کار برای نرم‌افزارهای بزرگ و پیچیده احتمال کشف خطا را به شدت پائین آورده و نیز هزینه را بالا می‌برد. خودکارسازی تولید نمونه آزمون از اهمیت بالایی برخوردار است. این اهمیت در مورد سیستم‌های ایمنی بحرانی بسیار قابل توجه است. در این سیستم‌ها هرگونه شکست منجر به صدمه یا از دست رفتن زندگی افراد جامعه می‌شود. معمولاً حدود ۵۰٪ از هزینه مصرفی در چرخه تولید نرم‌افزار، به این بخش تعلق دارد [۴-۳]. آزمون نرم‌افزار می‌تواند در هر مرحله از چرخه حیات تولید نرم‌افزار صورت گیرد، با این تفاوت که در هر مرحله ماهیت متفاوتی خواهد داشت. دنباله‌های آزمونی که به‌عنوان ورودی فرآیند آزمون در نظر گرفته می شوند بسیار بزرگ هستند و عملاً بررسی تمامی آن‌ها امکان‌پذیر نیست. با حذف نمونه آزمون‌های افزونه می‌توان به زیرمجموعه‌ای از دنباله‌های آزمون رسید که ضمن پوشش کامل مشخصات، تعداد خطای بیش‌تری را نیز آشکار کند.

روش ساختاری و روش عملکردی دو دسته‌بندی مهم از روش‌های آزمون هستند. در آزمون ساختاری واحد نرم‌افزار به شکل یک «جعبه سفید» در نظر گرفته می‌شود. در این روش انتخاب نمونه آزمون‌ها بر مبنای کد پیاده‌سازی سیستم و به هدف اجرای عبارات خاص، شاخه‌های برنامه یا مسیرهای خاص است. در آزمون عملکردی سیستم به‌عنوان یک «جعبه سیاه» در نظر گرفته می‌شود. انتخاب نمونه آزمون‌ها در این روش بر مبنای نیازمندی‌ها یا طراحی خصوصیات نرم‌افزار است[۵]. مهم‌ترین فایده این روش این است که آزمون‌گر به هیچ اطلاعاتی در مورد پیاده‌سازی نیاز ندارد[۶].

تاکنون تکنیک‌های مختلفی برای تولید نمونه آزمون بهینه ارائه شده است. در این میان الگوریتم‌های فرامکاشفه‌ای سهم به سزایی را دارند. روش‌ها و الگوریتم‌های بهینه‌سازی به دو دسته الگوریتم‌های دقیق و الگوریتم‌های تقریبی تقسیم‌بندی می‌شوند. الگوریتم‌های دقیق قادر به یافتن جواب بهینه به‌صورت دقیق هستند اما در مورد مسائل بهینه‌سازی سخت کارایی ندارند و زمان حل آن‌ها در این مسائل به‌صورت نمایی افزایش می‌یابد. الگوریتم‌های تقریبی قادر به یافتن جواب‌های خوب (نزدیک به بهینه) در زمان حل کوتاه برای مسائل بهینه سازی سخت هستند. این الگوریتم‌ها به سه دسته الگوریتم‌های مکاشفه‌ای و فرامکاشفه‌ای و فوق‌مکاشفه‌ای بخش‌بندی می‌شوند. دو مشکل اصلی الگوریتم‌های مکاشفه‌ای، قرار گرفتن در بهینه‌های محلی و ناتوانی آن‌ها برای کاربرد در مسائل گوناگون است. الگوریتم‌های فرامکاشفه‌ای یکی از انواع الگوریتم‌های بهینه سازی تقریبی هستند که دارای راهکارهای برون‌رفت از بهینه محلی می‌باشند و قابل کاربرد در طیف گسترده‌ای از مسائل هستند[۷]. رده‌های گوناگونی از این نوع الگوریتم در دهه‌های اخیر توسعه یافته است[۸].

یکی از معیارهای سنجش استراتژی‌های مختلف در تولید دنباله آزمون کمینه پوشش آرایه[2] (CA) است. پوشش آرایه که با نماد CA (N: t, k, v) نشان داده می شود، یک آرایه با تعداد N سطر و k ستون است که هر یک از پارامترهای ورودی v مقدار را می‌پذیرد. در این آرایه هر زیرآرایه t ستونی باید پوشش کامل را برای v مقدار ارضاء کند. هدف اصلی یافتن N با کم‌ترین مقدار ممکن و در کم‌ترین زمان است. این مسئله یک مسئله NP[3] سخت (مسائلی هستند که تاکنون راه‌حل سریع با مرتبه زمانی چندجمله‌ای برای آن‌ها یافت نشده است[۹]) است و برای تولید این آرایه استراتژی‌های مختلفی وجود دارد که این استراتژی‌ها به دو دسته کلی زیر تقسیم می‌شوند[۱۰]:

- استراتژی‌های مبتنی بر محاسبات محض[4]
- استراتژی‌های مبتنی بر هوش مصنوعی[5]

دسته اول شامل استراتژی های GTWay [۱۱]، AETG، [۱۲]Maetg، [۱۳] IPO، [۱۴] IPOG، [۱۵] MC-IPO، [۱۶] و Jenny [۱۷] و مانند آن است. این دسته از استراتژی‌ها همیشه تعداد ثابتی برای یک پیکربندی مشخص تولید می‌کنند و دسته دوم شامل الگوریتم شبیه سازی تبرید [۱۸]، الگوریتم جستجوی مورچگان [۱۹]، الگوریتم بهینه‌سازی توده ذرات [۲۰] و الگوریتم جستجوی هارمونیک و نظایر آن است. ازآنجایی که الگوریتم های مبتنی بر هوش مصنوعی بر پایه تصادف[6] می‌باشند لذا برخلاف استراتژی‌های مبتنی بر محاسبات محض قطعیتی برای این الگوریتم‌ها وجود ندارد و معمولاً بهترین مقدار را در چند تکرار در نظر می‌گیرند.

در بین الگوریتم های مبتنی بر محاسبات محض، الگوریتم های Jenney، Tconfig، PICT، TVG، IPOG قادر به تولید دنباله آزمون تا مقدار T=۶ با سرعت بالا می‌باشند که سایز مستخرج‌شده از استراتژی‌های یادشده قابل مقایسه با سایر الگوریتم‌ها نیست. از این دسته الگوریتم TVG در T=۲ نتایج قابل قبولی تولید می کند و الگوریتم ITCH در T=۲ برای بسیاری از پارامترها و در T=۳ برای پارامتر های کوچک‌تر از ۸ نتایج خوبی را تولید خواهد کرد. اما این استراتژی فقط تا T=۴ قادر به تولید دنباله آزمون خواهد بود.

قوی‌ترین استراتژی‌های مبتنی بر هوش مصنوعی در این حوزه الگوریتم SA، استراتژی PSTG [۱۰] و استراتژی HSS [۲۱] می‌باشند. الگوریتم SA برای T=۲ و T=۳ بهترین نتایج ممکن را استخراج می‌کند و برای T≥۴ نتایجی در دسترس نیست. استراتژی PSTG قادر خواهد بود تا T=۶ دنباله آزمون بهینه را تولید کند که در T=۴، T=۵ و T=۶ بهترین نتیجه را تولید می‌کند، ولی برای T≥۷ قادر به تولید دنباله آزمون نخواهد بود. دیگر الگوریتم قدرتمند مبتنی بر هوش مصنوعی استراتژی HSS است. این استراتژی قادر به تولید دنباله آزمون تا T=۱۵ خواهد بود که در بسیاری از پیکربندی‌ها بهترین نتایج را تولید می‌کند.

ایرادی که به این استراتژی وارد است زمان طولانی برای تولید دنباله آزمون است.

این پژوهش با ترکیب الگوریتم جستجوی تپه‌نوردی و الگوریتم جستجوی خفاش، دنباله آزمون بهینه را تولید می‌کند. دنباله آزمون تولیدشده از لحاظ اندازه آرایه از سایر استراتژی‌های موجود بهتر عمل می‌کند و از لحاظ زمان تولید نیز دارای خروجی مطلوبی است.

در ادامه مقاله در بخش ۲ با عنوان تعاریف و مفاهیم بنیادی به تشریح مفاهیمی مانند آرایه متعامد، پوشش آرایه و الگوریتم تپه‌نوردی و الگوریتم خفاش پرداخته می‌شود. در بخش ۳ طرح مسئله و چرایی انجام پژوهش بررسی می‌گردد. بخش ۴ به تشریح کامل راه‌حل پیشنهادی خواهد پرداخت و نتایج راهکار پیشنهادی در بخش ۵ نشان داده شده است. در ادامه این بخش نتایج مورد ارزیابی قرار می‌گیرد و با سایر استراتژی‌های موجود مقایسه می‌گردد. در بخش ۵ نیز به جمع‌بندی و نتیجه‌گیری مختصری در خصوص پژوهش اشاره می‌شود و ویژگی‌های روش پیشنهادی و کارهایی که می‌توان در آینده بر روی آن متمرکز شد مطرح شده است.

۲- تعاریف و مفاهیم بنیادی

در این بخش ابتدا به تعاریف مهمی که در حوزه آزمون نرم‌افزار مطرح هستند پرداخته می‌شود. سپس مفاهیم مربوط به الگوریتم‌های فرامکاشفه‌ای تپه‌نوردی و خفاش شرح داده می‌شود.

۲-۱- آزمون آرایه متعامد[۷]

دامنه ورودی بسیاری از برنامه‌های کاربردی محدود است. یعنی تعداد پارامترهای ورودی، کوچک و مقادیری که هر یک از پارامترها به خود می‌گیرند دارای مرز مشخصی هستند. هنگامی که این اعداد بسیار کوچک باشند (مثلا" ۳ پارامتر ورودی که هر یک ۲ مقدار (۰ و ۱) مجزا می‌گیرند)، می‌توان تمام حالت‌های ورودی را در نظر گرفت و پردازش دامنه ورودی را به‌طور جامع، مورد آزمایش قرار داد (جدول ۱) ولی با رشد تعداد مقادیر ورودی و تعداد مقادیر مجزا جهت هر عنصر داده‌ای، آزمون جامع غیرعملی و امکان‌ناپذیر می‌شود. آزمون آرایه متعامد را می‌توان در مورد مسائلی به کاربرد که در آن‌ها دامنه ورودی نسبتا" کوچک است و برای اجرای آزمون جامع بیش از حد بزرگ است، معیار T-ستونی در یافتن خطاهای مرتبط با خطاهای ناحیه‌ای مفید واقع می‌شود. خطای ناحیه‌ای به گروهی از خطاها اطلاق می‌شود که به منطق نادرست در نقطه‌ای از یک برنامه مربوط می‌شوند.

برای نشان دادن اختلاف میان معیار پوشش T-ستونی و روش سنتی "یک عنصر ورودی در هر نوبت"، سیستمی را در نظر بگیرید که دارای سه ورودی z و y و x که هر یک از این عناصر ورودی دارای دو مقدار (۰ و ۱) هستند که $۲^۳$=۸ نمونه آزمون متفاوت امکان‌پذیر است (جدول ۱). حال اگر مجموع آزمون پوششی سه پارامتر بولین فوق با روش T-ستونی محاسبه گردد، حاصل جدول ۲ است. در این معیار پوشش به‌جای در نظر گرفتن تمام ستون‌ها، ترکیب ۲ ستون در نظر گرفته می شود (ستون‌های ۱ و ۲، ستون‌های۱ و ۳، ستون‌های ۲ و ۳). درصورتی‌که در تمامی این جفت ستون‌ها مقادیر ۰۰، ۰۱، ۱۰ و ۱۱ وجود داشته باشد به معنای پوشش کامل با معیار ۲-ستونی خواهد بود [۲۲].

جدول ۱: حالات آزمون سه پارامتر بولین [۲۲]

K	A	B	C
۱	۰	۰	۰
۲	۰	۰	۱
۳	۰	۱	۰
۴	۰	۱	۱
۵	۱	۰	۰
۶	۱	۰	۱
۷	۱	۱	۰
۸	۱	۱	۱

تعریف ۱: CA (N: t, k, v) یک آرایه N*k بر روی v نماد است که هر زیرآرایه N*t از آن، تمام اشکال t تایی از آن v نماد را حداقل یکبار پوشش می‌دهد. در چنین آرایه‌ای به t قوه گفته می شود [۲۳، ۲۴] که تعداد کل پوشش‌ها از رابطه (۱) به دست می‌آید.

جدول ۲: دنباله آزمون ۲-ستونی با ورودی ۳ پارامتر بولین [۲۲]

K	A	B	C
۱	۰	۰	۰
۲	۰	۱	۱
۳	۱	۰	۱
۴	۱	۱	۰

به‌عنوان‌مثال (۴:۲،۳،۲) CA یک آرایه ۳×۴ بر روی دو نماد (۰،۱) که هر زیر آرایه ۲ × ۴ از آن تمام حالت‌های ممکن {۰،۰}، {۱،۰}، {۰،۱} و {۱،۱} را حداقل یکبار پوشش می‌دهد. همان‌طوری که در جدول ۲ ملاحظه می شود تعداد سطرها از ۸=$۲^۳$ به ۴ سطر کاهش پیدا می‌کند.

$$CA = \binom{k}{t} v^t \quad (۱)$$

۲-۲- الگوریتم جستجوی تپه‌نوردی

الگوریتم جستجوی تپه‌نوردی از ابتدایی‌ترین تکنیک‌های جستجوی محلی است. الگوریتم جستجوی تپه‌نوردی، همانند شکل ۱ حلقه‌ای است که در جهت افزایش مقدار حرکت می‌کند (به‌طرف بالای تپه). وقتی به "قله‌ای" رسید که هیچ همسایه‌ای از آن بلندتر نیست، خاتمه می‌یابد. این الگوریتم، درخت جستجو را نگه‌داری نمی‌کند. لذا گره داده فعلی فقط باید حالت و مقدار تابع هدف را نگه‌داری کند. تپه‌نوردی به همسایه‌های حالت فعلی نگاه می‌کند. این سیاست ساده دارای سه اشکال است:

بیشینه محلی[۸]: قله‌ای است که بلندتر از هر یک از همسایه‌هایش است، اما از بیشینه سراسری کوتاه‌تر است. الگوریتم‌های تپه‌نوردی که به مجاورت بیشینه محلی می‌رسند، به‌طرف بالا و به‌سمت قله حرکت می‌کنند و از آن پس متوقف می‌شوند.

فلات[۹]: فلات می‌تواند یک بیشینه محلی مسطح باشد که هیچ خروجی به‌سمت بالا وجود ندارد، که از طریق آن پیشروی کند.

```
function HILL-CLIMBING( problem) returns a solution state
  inputs: problem, a problem
  static: current, a node
  next, a node
  current <— MAKE-NODE(INITIAL-TATE[problem])
loop do
  next'— a highest-valued successor of current
  if VALUE[next] < VALUE[current] then return current
    current *—next
end
```

شکل ۱: شبه‌کد الگوریتم تپه‌نوردی [۲۵]

برآمدگی‌ها[۱۰]: برآمدگی‌ها منجر به دنباله‌ای از بیشینه‌های محلی می‌شوند که عبور از آن برای الگوریتم‌های حریصانه دشوار است.

در هر یک از این موارد، الگوریتم به نقطه‌ای می‌رسد که پیشروی از آن ممکن نیست. درصورت رخ دادن هر یک از حالت‌های فوق می‌توان الگوریتم را از یک نقطه متفاوت شروع کرد. این راه‌اندازی مجدد تصادفی الگوریتم[۱۱] تپه‌نوردی روال را به‌سمت مقادیر جدید هدایت می‌کند و تا زمانی‌که به نتیجه دلخواه برسد و یا تعداد تکرار حلقه به حداکثر برسد الگوریتم ادامه خواهد داشت[۲۵].

۲-۳- الگوریتم جستجوی خفاش

یکی از الگوریتم‌های الهام‌گرفته از طبیعت الگوریتم جستجوی خفاش است. توانایی خفاش‌های کوچک در انعکاس صدا این موجودات را قادر می‌سازد که حتی در تاریکی کامل هم طعمه خود را پیدا کنند و بین انواع مختلف حشرات تفاوت قائل شوند. خفاش‌های کوچک از یک نوع سونار (دستگاه کاشف زیردریایی بوسیله امواج صوتی) که انعکاس صدا نامیده می‌شود برای تشخیص طعمه، اجتناب از موانع و رد شدن از شکاف لانه استفاده می‌کنند. این خفاش‌ها یک پالس، صدای بسیار بلند را منتشر می‌کنند و به اکو صدای اشیاء اطراف گوش می‌دهند. پالس آن‌ها متفاوت است که با استراتژی شکار آن‌ها در ارتباط است. بسیاری از خفاش‌ها از فرکانس با طول موج کوتاه استفاده می‌کنند درحالی‌که دیگران در اغلب موارد از سیگنال‌های با فرکانس ثابت برای انعکاس صدا استفاده می‌کنند. پهنای باند سیگنال‌ها متفاوت هستند و به گونه خفاش‌ها بستگی دارد. محدوده فرکانس برای بسیاری از گونه های خفاش‌ها بین ۲۵khz تا ۱۰۰khz است. هر چند بعضی از گونه‌ها می‌توانند فرکانس‌های بالاتر تا ۱۵۰khz منتشر کنند. هر انفجار مافوق صوت ممکن است به‌طور معمول ۵ تا ۲۰ میلی‌ثانیه طول بکشد و خفاش ها حدود ۱۰ تا ۲۰ انفجار صوتی در هر ثانیه منتشر می‌کنند. در هنگام شکار زمانی که در نزدیکی شکار هستند سرعت انتشار پالس می‌تواند تا حدود ۲۰۰ پالس در هر ثانیه برسد. اگر ما برخی از ویژگی انعکاس صدای خفاش‌ها را به‌صورت ایده‌آل درآوریم، می‌توانیم از الگوریتم‌های مختلف خفاش الهام گرفته یا الگوریتم خفاش را توسعه دهیم. برای سادگی، از قوانین تقریبی زیر استفاده می‌شود:

همه خفاش‌ها از انعکاس صدا برای تعیین فاصله استفاده می‌کنند و آن‌ها نیز تفاوت بین مواد غذایی/شکار و موانع پیش رو را به‌صورت خارق‌العاده‌ای می‌دانند.

پرواز خفاش‌ها به‌صورت تصادفی با سرعت V_i و در مکان X_i با فرکانس ثابت fmin و طول موج مختلف λ و بلندی صدا A_0 برای جستجوی طعمه صورت می‌گیرد و می‌توانند به‌طور خودکار طول موج (یا فرکانس) امواج پخش‌شده خود را تنظیم کنند، و نرخ امواج انتشار $R2$ با توجه به نزدیکی هدف خود را تنظیم کنند.

اگرچه بلندی صدا می‌تواند در بسیاری جهات متفاوت باشد، فرض خواهد شد که بلندی صدا از A_0 با مقادیر بزرگ (مثبت) به کمینه مقدار ثابت (A_{min}) قابل تغییر است.

برای خفاش ها عموماً بازه بیش از چند متر نیست. نرخ موج می‌تواند به‌راحتی در بازه صفر و یک باشد که صفر یعنی هیچ موجی اصلا وجود ندارد و ۱ یعنی بیش‌ترین نرخ انتشار موج. بر اساس این فرضیات و نظریات، گام‌های پایه‌ای در الگوریتم خفاشی در شکل ۲ آورده شده است. در شبیه‌سازی‌ها معمولاً از خفاش‌های مجازی استفاده می شود. باید قوانینی تعریف شود که چگونه مکان‌ها و سرعت ها در یک جستجوی مکان بعدی تغییر کنند. راه‌حل‌ها با X_i^t و V_i^tهای جدید در بازه زمانی T از رابطه (۲)، رابطه (۳) و رابطه (۴) به دست می‌آیند[۲۶].

$$fi = fmin + (fmax - fmin)B \tag{۲}$$

$$V_i^t = V_i^{t-1} + (X_i^t - X_*)fi \tag{۳}$$

$$X_i^t = X_i^{t-1} + V_i^t \tag{۴}$$

۳- طرح مسئله

سیستم‌های نرم‌افزاری باید دارای قابلیت اعتماد بالایی باشند. برای رسیدن به این هدف، نرم‌افزار باید با نمونه‌های آزمون کافی آزمایش شود. آزمون نرم‌افزار یک فاز حیاتی در توسعه نرم‌افزار است. اغلب، تعداد نمونه‌های آزمون بسیار زیاد است که بررسی تمامی آن‌ها زمان‌بر است. با حذف نمونه های آزمون افزونه می‌توان به زیرمجموعه‌ای از دنباله های آزمون رسید که پوشش کامل را دربر بگیرد. آزمون‌گر به مجموعه‌ای از نمونه آزمون نیاز دارد که اهداف آزمون را ارضاء کند.

```
Objective function f(x),  x = (x1, ..., xd)T
Initialize the bat population xi (i = 1, 2, ..., n) and vi
Define pulse frequency fi at xi
Initialize pulse rates ri and the loudness Ai
while (t <Max number of iterations)
  Generate new solutions by adjusting frequency,
  and updating velocities and locations/solutions [equations (2) to (4)]
  if (rand > ri)
    Select a solution among the best solutions
    Generate a local solution around the selected best solution
  end if
  Generate a new solution by flying randomly
  if (rand < Ai & f(xi) < f(x*))
    Accept the new solutions
    Increase ri and reduce Ai
  end if
  Rank the bats and find the current best x*
end while
     Postprocess results and visualization
```

شکل ۲: شبه کد الگوریتم خفاش [۲۶].

این مجموعه از نمونه های آزمون به‌عنوان دنباله آزمون شناخته می شوند. در هنگام تولید نمونه آزمون ممکن است نمونه آزمونی حذف شود که باعث ارضاء نشدن هدف آزمون گردد. بنابراین دنباله آزمون تولید شده باید زیرمجموعه‌ای از دنباله‌های آزمون اصلی باشد که هدف آزمون را ارضاء کند. در این پژوهش راهکار ترکیبی با استفاده از الگوریتم های فرامکاشفه‌ای ارائه خواهد شد. روش پیشنهادی برای انتخاب دنباله آزمون از استراتژی چندستونی و ترکیب الگوریتم‌های تپه‌نوردی-خفاش استفاده می‌کند که مجموعه آزمون کمینه را با بیش‌ترین پوشش حالت‌های آزمون به دست می‌آورد.

۴- راه‌حل پیشنهادی

راهکار جدیدی که در این پژوهش ارائه شده است استفاده ترکیبی الگوریتم‌های جستجوی خفاش و تپه‌نوردی در حوزه تولید دنباله آزمون است. این پژوهش به کمک استراتژی پوشش آرایه (CA) سعی در تولید دنباله آزمون با کم‌ترین تعداد را دارد. در این راستا جهت تولید دنباله آزمون کمینه با تغییرات در الگوریتم خفاش و الگوریتم تپه‌نوردی و ترکیب این دو الگوریتم، دنباله آزمون بهینه را تولید می‌کند. یکی از زمان‌برترین مراحل تولید دنباله آزمون تابع ارزیابی است که برابر با پوشش‌های جدیدی است که در دنباله آزمون وجود ندارد. هنگامی‌که تعداد کل پوشش‌های جدید به رابطه (۱) بر سد به معنای پوشش کامل است و الگوریتم در آن زمان پایان می‌پذیرد و درصورتی‌که تعداد تکرار الگوریتم پایان یابد و پوشش به حدنصاب نرسد الگوریتم قادر به تولید دنباله آزمون نخواهد بود.

مراحل اجرای الگوریتم پیشنهادی در شکل ۳ نشان داده شده است. در الگوریتم تولید CA (N: t, k, v) هر ذره شامل دو قسمت داده و وزن است. داده‌ها از یک آرایه k عنصری تشکیل شده است که مقادیر عناصر آن بین ۰ تا v-۱ است و وزن آن از تابع ارزیابی به دست می‌آید. در گام اول ازآنجایی‌که وزن سطر اول دنباله آزمون برای تمامی مقادیر $X_i(t)$ برابر C(k, t) خواهد بود، لذا سطر اول به‌طور تصادف انتخاب می‌گردد. الگوریتم پیشنهادی دارای دو حلقه اصلی است که اولین حلقه تعداد کل تکرار را مشخص می‌کند. در پیاده‌سازی‌های معمول پس از پایان این حلقه به یک جواب نهایی برای مسئله خواهیم رسید اما ماهیت تولید دنباله آزمون به‌گونه‌ای است که در هر تکرار یک نمونه آزمون از دنباله آزمون انتخاب می‌شود و این نمونه آزمون هیچگونه تأثیری بر مراحل بعدی اجرای حلقه نخواهد داشت، به همین دلیل در ابتدای این حلقه بهترین مقدار (best) به‌طور تصادف انتخاب خواهد شد. دیگر حلقه در الگوریتم پیشنهادی به تعداد جمعیت ذرات تکرار خواهد شد که این مقدار با توجه به پارامترهای ورودی CA بین ۸۰ تا ۵۰۰ متغیر است که در داخل این حلقه پس از تولید مقدار جدید $X_i(t)$ و ایجاد همسایه برای best وزن هر یک محاسبه می‌گردد و best با بهترین مقدار عوض خواهد شد. در پایان این حلقه best به دنباله آزمون اضافه می‌گردد و تا زمانی‌که پوشش کامل شود الگوریتم ادامه خواهد داشت. تابع تولید همسایه یک تابع ساده و نحوه عمل‌کرد آن به‌صورتی است که ۲ یا ۳ عنصر از k عنصر را به‌صورت تصادفی انتخاب و برای هریک مقداری تصادفی بین ۰ و v-۱ انتخاب می‌کند. ازآنجایی‌که این تغییر بر روی بهترین مقدار صورت خواهدگرفت لذا تابع بسیار مؤثر بوده و اصلی‌ترین دلیل برتری آن نسبت به دیگر پژوهش‌ها است.

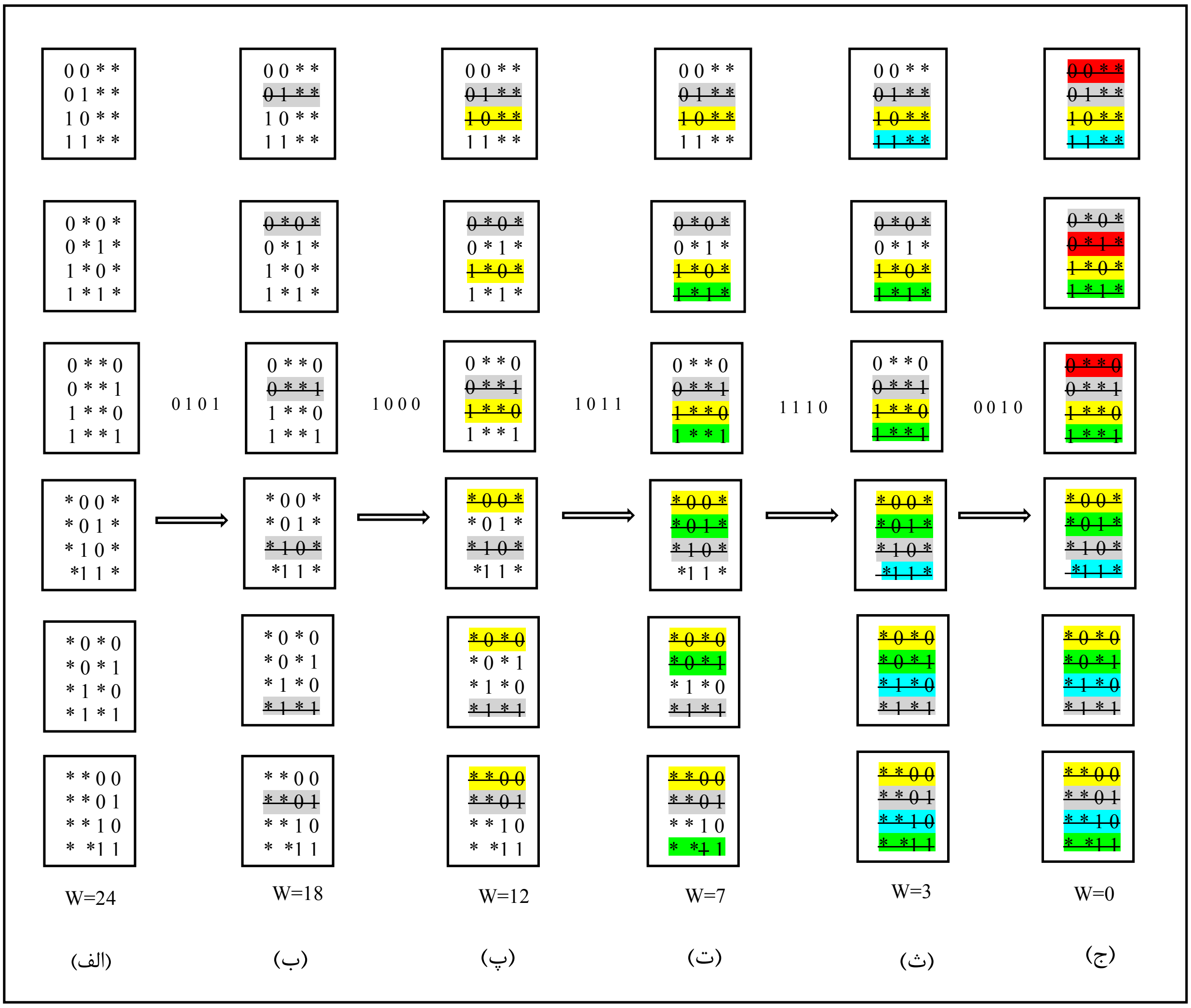
```
Initialize Parameters
TS=Ø
Randomaly initialize Best
TS=TS U best
While (t<number of iteration and max_wieght>0)
    Randomaly initialize Best
    Best.weight=max_weight
    For i=1 to population size
        Randomaly initialize particle xi(t) and vi(t)=0
        updating velocities [equations (2)]
        xi(t)=best+ vi(t)
        compute the weight of coverage for xi(t)
        for j=1 to neighborhoods count
            generate new neighberhood
            compute the weight of coverage for neighberhood
            if  weight(best)<weight(neighberhood)
                best= neighberhood
        if  weight(best)<weight(xi(t))
            best= xi(t)
    TS=TS U best
    Max_weight=max_weight-weight(best)
End while
```

شکل۳: ترکیب الگوریتم‌های خفاش و تپه‌نوردی برای تولید دنباله

برای تشریح بیش‌تر راهکار پیشنهادی مراحل الگوریتم بر روی مثال CA (N:۲،۴،۲) پیاده‌سازی می‌شود. ابتدا تعداد کل پوشش از رابطه (۱) محاســبه می‌گردد که این مقدار مطابق شــکل ۴-الف برابر ۲۴ خواهد بود. درگام بعدی به‌صــورت تصــادفی نمونه آزمون (TC=۰۱۰۱) تولید خواهد شد. که تعداد پو شش‌های جدید برابر ۶ نمونه آزمون ا ست. این مقدار از تطبیق نمونه آزمون (شــکل ۴-ب) با **۰۱ ، *۰*۰، ۱**۰، *۱۰*، ۱*۱* و ۰۱** به دســت می‌آید ســپس نمونه‌های آزمون پوشــش داده‌شــده از لیســت پوشــش حذف و مقدار آن نیز از تعداد کل پوشــش کســر می‌شــود و نمونه آزمون به دنباله آزمون اضــافه می‌گردد (TS=۰۱۰۱). تعداد پوشـش‌های باقی‌مانده در این مرحله به ۱۸ خواهد رسـید. در گام بعد برای تمامی ذرات الگوریتم، مقدار تصـادفی تولید می‌گردد و پو شش باقی‌مانده آن‌ها نیز برابر ۱۸ می شود. مقدار هریک از ذرات با تغییر مختصر در رابطه (۲)، رابطه (۳) و رابطه (۴) تولید می‌گردد که این تغییر از [۲۷] گرفته شده است. پس از آن در هر تکرار ۲ همسایه برای بهترین مقدار به دست می‌آید. پس از تولید همسایه وزن ذره جدید و همسایه‌ها محا سبه می‌گردد. در صورتی‌که وزن آن‌ها بیش‌تر از بهترین مقدار بود آن را جایگزین می‌کند. مطابق شکل ۳ دفعات تکرار این حلقه برابر جمعیت ذرات است. پس از اتمام این حلقه به TC=۱۰۰۰ خواهد رســید که w=۶ اســت. در ادامه مقدار وزن کل (W=۱۸-۶)، لیســت پوشــش (شــکل ۴-پ) و لیســت دنباله آزمون (TS=TSU۱۰۰۰) بروز می‌شــود. گام بعدی TC=۱۰۱۱ (شــکل ۴-ت) با وزن برابر ۵ انتخاب می‌گردد و به لیســت اضــافه می‌شــود. مرحله بعد TC=۱۱۱۰ با وزن ۴ (شــکل ۴-ث) و درآخرین مرحله TC=۰۰۱۰ با وزن ۳ (شــکل ۴-ج) انتخاب می‌گردد و وزن تعداد پوشش باقی‌مانده صفر می‌شود و الگوریتم پایان می‌پذیرد.

۵- ارزیابی

ارزیابی اســتخراج دنباله آزمون به دو دســته کلی اندازه آرایه[۱۲] و زمان[۱۳] ا ستخراج دنباله آزمون تق سیم می شود. برر سی زمان تولید دنباله آزمون بــه دو دلــیل امکــان‌پــذیــر نیــست. بــرخــی از استــراتــژی‌هــا (SA، ACA،Density و ParaOrder) در دســترس عموم قرار نـدارنـد، ازاین‌رو باید به‌ناچار از نتایج منت شر شده در ن شریات ا ستفاده شود که به دلیل تفاوت در بستر سخت‌افزاری و سیستم‌عامل، مقایسه عادلانه بسیار دشــوار خواهد بود. و در مقابل اندازه آرایه وابســته به محیط اجرایی نخواهد بود و تنها دلیل مؤثر بر این معیار مراحل اجرای الگوریتم اســت. در این ارزیابی نمایش برجســته[۱۴] نمایانگر کم‌ترین مقدار در پیکربندی مربوطه اســت. واژه NA[۱۵] به مفهوم دســترســی به دنباله آزمون در مدت‌زمان بیش از یک روز یا عدم انت شار در ن شریات ا ست و NS[۱۶] عدم توانـایی اســتراتژی در تولیـد دنبـالـه آزمون در پیکربنـدی مربوطـه است[۱۰]. مشخصات بستر سخت‌افزاری برای اجرای روش پیشنهادی ع بارت‌ا ند از: Windows 7، i7QM CPU 2.20GHz core™ و RAM 6GB. و کدنویسی آن در محیط MATLAB R2013b انجام گرفته است. در را ستای ارزیابی راهکار پی شنهادی سعی خواهد شد نتایج ا ستخراج شــده از الگوریتم تا حد امکان با تمامی الگوریتم‌ها و اســتراتژی‌های موجود اعم از استراتژی‌های مبتنی بر محا سبات محض و استراتژی‌های مبتنی بر هوش م صنوعی مقایسه گردد و برتری الگوریتم حداقل در یک یا چند پیکربندی نسبت به استراتژی‌های دیگر نشان داده شود.



شکل ٤: مراحل اجرای الگوریتم پیشنهادی بر روی پیکربندی **CA (N ;۲،٤،۲)**

برای ارزیابی الگوریتم پیشنهادی از نتایج منتشرشده در [۱۰، ۲۸] استفاده خواهد شد که شامل چند پیکربندی از CA (N:t, p, v) است. در پیکربندی نخست ۴≥T≥۲ و ۱۲≥k≥۳ که هر یک از پارامترها دارای سه مقدار (v=۳) می‌باشند. در جدول ۳ و جدول ۴ الگوریتم پیشنهادی با استراتژی‌های مبتنی بر محاسبه محض مانند Jenney، Tconfig، ITCH، TVG، CTE-XL، IPOG و IPOG-D و استراتژی مبتنی بر هوش مصنوعی PSTG مقایسه می‌گردد. همان‌طور که در جدول ۳ نشان داده شده است برای T=۲ و k=۳ استراتژی‌های Jenney، ITCH، PSTG و الگوریتم پیشنهادی دنباله آزمون با اندازه ۹ نمونه آزمون را تولید خواهند کرد که از سایر استراتژی‌های موجود بهتر عمل می‌کنند و برای k=۴ استراتژی‌های ITCH، PSTG و الگوریتم پیشنهادی بهترین دنباله آزمون را تولید می‌کند. در k=۵ و k=۶ استراتژی PSTG و الگوریتم پیشنهادی بهترین عملکرد را دارند. زمانی که تعداد پارامترها به ۷ برسد راهکار پیشنهادی با تولید دنباله آزمون با اندازه ۱۴ بهترین عملکرد را در بین استراتژی‌های موجود دارد، اما برای ۱۲≥k≥۸ استراتژی ITCH بهترین دنباله آزمون را استخراج می‌کند. در قسمت دوم جدول ۳ به بررسی T=۳ برای ۱۲≥k≥۴ پرداخته می شود. ابتدا به k=۴ اشاره خواهد شد. در این پیکربندی استراتژی‌های ITCH، IPOG-D و PSTG بهترین نتایج را تولید می‌کنند. در ۶≥k≥۵ الگوریتم پیشنهادی قادر به تولید دنباله آزمون با کم‌ترین تعداد نمونه آزمون خواهد بود و در ۸≥k≥۷ الگوریتم ITCH کم‌ترین مقادیر را تولید می‌کند. زمانی که تعداد پارامترها از ۸ بیش‌تر شود رقابت فقط بین الگوریتم پیشنهادی و استراتژی PSTG خواهد بود که برای k=۹ و k=۱۲ الگوریتم پیشنهادی بهتر عمل خواهد کرد. در k=۱۰ و k=۱۱ نتایج یکسانی تولید می‌گردد. در سومین قسمت جدول ۳ نیز رقابت اصلی بین دو استراتژی مذکور خواهد بود که در۷≥k≥۵ راهکار پیشنهادی عملکرد بهتری را دارد و در ۱۰≥k≥۸ نتایج یکسان تولید می‌گردد. استراتژی PSTG در ۱۲≥k≥۱۱ نتایج بهتری را ارائه می‌کند. در این پیکربندی استراتژی CTE-X دیگر قادر به تولید دنباله آزمون نخواهد بود و سایر استراتژی‌ها نیز نتایج قابل‌قبولی را تولید نمی‌کنند.

جدول ۳: تعداد نمونه آزمون **CA (N: t, k, v)** برای ۱۲≥k≥۳ سه مقداره

T	K	Jenny	TConfig	ITCH	PICT	TVG	CTE-XL	IPOG-D	IPOG	PSTG	HC-BAT

۲	۳	۹	۱۰	۹	۱۰	۱۰	۱۰	۱۵	۱۱	۹	۹
	٤	۱۳	۱۰	۹	۱۳	۱۲	۱٤	۱۵	۱۲	۹	۹
	٥	۱٤	۱٤	۱۵	۱۳	۱۳	۱٤	۱۵	۱٤	۱۲	۱۲
	٦	۱۵	۱۵	۱۵	۱٤	۱۵	۱٤	۱۷	۱۵	۱۳	۱۳
	۷	۱٦	۱۵	۱۵	۱٦	۱۵	۱٦	۱۸	۱۷	۱۵	۱٤
	۸	۱۷	۱۷	۱۵	۱٦	۱۵	۱۷	۱۸	۱۷	۱۵	۱۵
	۹	۱۸	۱۷	۱۵	۱۷	۱۵	۱۸	۲۰	۱۷	۱۷	۱٦
	۱۰	۱۹	۱۷	۱۵	۱۸	۱٦	۱۸	۲۰	۲۰	۱۷	۱۷
	۱۱	۱۷	۲۰	۱۵	۱۸	۱٦	۲۰	۲۱	۲۰	۱۷	۱۷
	۱۲	۱۹	۲۰	۱۵	۱۹	۱٦	۲۰	۲۱	۲۰	۱۸	۱۷
۳	٤	۳٤	۳۲	۲۷	۳٤	۳٤	۳٤	۲۷	۳۹	۲۷	۲۸
	٥	٤۰	٤۰	٤٥	٤۳	٤۱	٤۳	٤۹	٤۳	۳۹	۳۸
	٦	٥۱	٤۸	٤٥	٤۸	٤۹	٥۲	٤۹	٥۳	٤٥	٤۳
	۷	٥۱	٥٥	٤٥	٥۱	٥٥	٥٤	٦۳	٥۷	٥۰	٤۹
	۸	٥۸	٥۸	٤٥	٥۹	٦۰	٦۳	٦۳	٦۳	٥٤	٥۳
	۹	٦۲	٦٤	۷٥	٦۳	٦٤	٦٦	۷۱	٦٥	٥۸	٥۷
	۱۰	٦٥	٦۸	۷٥	٦٥	٦۸	۷۱	۷۱	٦۸	٦۲	٦۲
	۱۱	٦٥	۷۲	۷٥	۷۰	٦۹	۷٦	۸۳	۷٦	٦٤	٦٤
	۱۲	٦۸	۷۷	۷٥	۷۲	۷۰	۷۹	۸۳	۷٦	٦۷	٦٦
٤	٥	۱۰۹	۹۷	۱٥۳	۱۰۰	۱۰٥	NS	NS	۱۱٥	۹٦	۹۳
	٦	۱٤۰	۱٤۱	۱٥۳	۱٤۲	۱۳۹		۲۳٤	۱۸۱	۱۳۳	۱۳۲
	۷	۱٦۹	۱٦٦	۲۱٦	۱٦۸	۱۷۲		NS	۱۸٥	۱٥٥	۱٥٤
	۸	۱۸۷	۱۹۰	۲۱٦	۱۸۹	۱۹۲		۳۸٤	۲۰۳	۱۷٥	۱۷٥
	۹	۲۰٦	۲۱۳	۳۰٦	۲۱۱	۲۱٥		NS	۲۳۸	۱۹٥	۱۹٥
	۱۰	۲۲۱	۲۳٥	۳۳٦	۲۳۱	۲۳۳		٤۹۸	۲٤۱	۲۱۰	۲۱۰
	۱۱	۲۳٥	۲٥۸	۳٤۸	۲٤۹	۲٥۰		NS	۲۷۲	۲۲۲	۲۲٥
	۱۲	۲٥۲	۲۷۲	۳۷۲	۲٦۹	۲٦۸		٦٥٦	۲۷٥	۲٤٤	۲٤٥

در ادامه ارزیابی به بررسی جدول ۴ پرداخته می‌شود. پیکربندی در این جدول به‌گونه‌ای است که تعداد پارامتر ثابت (k=۷) و مقادر پارامترها متغیر خواهد بود که این مقادیر بین ۲ تا ۵ است. در این قسمت نیز همانند بخش ارزیابی قبل به بررسی استراتژی‌های Jenney، TConfig، ITCH، TVG، CTE-XL، POG، IPOG-D و PSTG پرداخته می‌شود. جدول ۴ شامل ۴ قسمت برای مقادیر ۲≤T≤۵ است. در قسمت اول (T=۲) برای v=۲ استراتژی ITCH، PSTG و راهکار پیشنهادی با تولید دنباله آزمون به اندازه ۶ نمونه آزمون بهترین عمل‌کرد را خواهند داشت. لازم به‌ذکر است که فقط در این مورد خاص الگوریتم پیشنهادی در صورتی قادر به تولید دنباله آزمون با این اندازه خواهد بود که تعداد جمعیت ذرات بسیار کوچک در نظر بگیریم با این کار الگوریتم در بعضی از تکرارها به نمونه آزمون با وزن بالا نخواهد رسید که این امر در این پیکربندی مفید واقع می‌شود. می‌توان نتیجه گرفت که همیشه انتخاب نمونه آزمون با بیش‌ترین وزن لزوما بهترین جواب برای مسئله نخواهد بود. در ادامه ارزیابی برای v=۳ نیز سه استراتژی فوق الذکر بهترین دنباله آزمون‌ها را تولید خواهند کرد. در v=۴ و v=۵ راهکار پیشنهادی بهترین دنباله آزمون را تولید می‌کند. در این پیکربندی استراتژی PSTG نیز نتایج قابل‌قبولی را تولید می‌کند. در قسمت دوم جدول ۴ به بررسی دنباله آزمون با T=۳ پرداخته می‌شود. الگوریتم ITCH در ۲≤v≤۵ بهترین مقادیر را استخراج می‌کند که در این پیکربندی الگوریتم پیشنهادی و استراتژی PSTG در v=۲ و v=۵ و همچنین استراتژی‌های Tconfig و IPOG-D در v=۴ مقادیر مشابه با استراتژی ITCH تولید خواهند کرد.

در قسمت بعدی بهترین نتایج را استراتژی PSTG و راهکار پیشنهادی تولید می‌کنند که راهکار پیشنهادی در ۲≤v≤۳ نسبت به استراتژی PSTG عملکرد بهتری دارد و در ۴≤v≤۵ استراتژی PSTG بهتر عمل می‌کند، که نتایج حاصله از دو استراتژی بسیار نزدیک است. در این پیکربندی استراتژی ITCH دیگر نتایج قابل‌قبولی را استخراج نمی‌کند و استراتژی‌های CTE-XL و IPOG-D در حالت NS قرار دارند و قادر به تولید دنباله آزمون نخواهند بود. در قسمت آخر جدول ۴ نتایج T=۵ مورد ارزیابی قرار می‌گیرد. در این پیکربندی برای v=۲ الگوریتم پیشنهادی و استراتژی PSTG با تولید ۵۳ نمونه آزمون بهترین نتیجه را در بین سایر استراتژی‌ها دارد.

جدول ۴: تعداد نمونه آزمون **CA (N: t, k, v)** برای ۲≤v≤۵ با **k=۷**

T	v	Jenny	TConfig	ITCH	PICT	TVG	CTE-XL	IPOG-D	IPOG	PSTG	HC-BAT
۲	۲	۸	۷	۶	۷	۷	۸	۸	۸	۶	۶
	۳	۱۶	۱۵	۱۵	۱۶	۱۵	۱۶	۱۸	۱۷	۱۵	۱۵
	۴	۲۷	۲۸	۲۸	۲۷	۲۷	۳۰	۲۸	۲۸	۲۶	۲۵
	۵	۳۸	۴۰	۴۵	۴۰	۴۲	۴۲	۵۲	۴۲	۳۷	۳۶
۳	۲	۱۴	۱۶	۱۳	۱۵	۱۵	۱۵	۱۴	۱۹	۱۳	۱۳
	۳	۵۱	۵۵	۴۵	۵۱	۵۵	۵۴	۶۳	۵۷	۵۰	۴۹
	۴	۱۲۴	۱۱۲	۱۱۲	۱۲۴	۱۳۴	۱۳۶	۱۱۲	۲۰۸	۱۱۶	۱۱۶
	۵	۲۳۶	۲۳۹	۲۲۵	۲۴۱	۲۶۰	۲۶۷	۲۹۲	۲۷۵	۲۲۵	۲۲۵
۴	۲	۳۱	۳۶	۴۰	۳۲	۳۱	NS	NS	۴۸	۲۹	۲۷
	۳	۱۶۹	۱۶۶	۲۱۶	۱۶۸	۱۶۷			۱۸۵	۱۵۵	۱۵۴
	۴	۵۱۷	۵۶۸	۷۰۴	۵۲۹	۵۵۹			۵۰۹	۴۸۷	۴۹۰
	۵	۱۲۴۸	۱۳۲۰	۱۷۵۰	۱۲۷۹	۱۳۸۵			۱۳۴۹	۱۱۷۶	۱۱۸۲
۵	۲	۵۷	۵۶	NS	۵۷	۵۹	NS	۹۶	۱۲۸	۵۳	۵۳
	۳	۴۵۸	۴۷۷		۴۵۲	۴۶۴		۷۳۵	۶۰۸	۴۴۱	۴۳۹
	۴	۱۹۳۸	۱۷۹۲		۱۹۳۳	۲۰۱۰		۳۱۱۸	۲۵۶۰	۱۸۲۶	۱۸۲۶
	۵	۵۸۹۵	NA		۵۸۱۴	۶۲۵۷		۹۴۶۵	۸۰۹۱	۵۴۷۴	۵۴۷۵

برای v=۳ الگوریتم پیشـــنهـادی بـا تعـداد ۳۳۹ نمونـه آزمون بهترین عملکرد را دارد و برای v=۴ استراتژی Tconfig و برای v=۵ استراتژی PSTG مقادیر کم‌تری را تولید می‌کنند. در این قسمت استراتژی ITCH و CTE-XL توانایی تولید دنباله آزمون را نخواهند داشـت و در حالت NS قرار دارند.

جدول ۵: تعداد نمونه آزمون در استراتژی‌های مبتنی بر هوش مصنوعی

configuration	HSS	SA	GA	ACA	PSTG	HC-BAT
CA(N:۲،۴،۳)	۹	۹	۹	۹	۹	۹
CA(N:۲،۱۳،۳)	۱۸	۱۶	۱۷	۱۷	۱۷	۱۸
CA(N:۲،۱۰،۱۰)	۱۵۵	NA	۱۵۷	۱۵۹	NA	۱۷۷
CA(N:۲،۱۰،۵)	۴۳	NA	NA	NA	۴۵	۴۳
CA(N:۳،۶،۳)	۳۹	۳۳	۳۳	۳۳	۴۵	۴۵
CA(N:۳،۶،۴)	۷۰	۶۳	۶۳	۶۳	۱۰۲	۱۰۵
CA(N:۳،۶،۵)	۱۹۹	۱۵۲	۱۲۵	۱۲۵	NA	۱۹۹
CA(N:۳،۷،۵)	۲۳۶	۲۰۱	۲۱۸	۲۱۸	۲۲۹	۲۲۲

در ادامه ارزیابی به بررســی نتایج منتشــرشــده در [۲۸] پرداخته می‌شـود. در این ارزیابی برتری راهکار پیشـنهادی نسـبت به اسـتراتژی PSTG نشـان داده می‌شـود. در جدول ۵ ارزیابی روی چند پیکربندی مجزا با اسـتراتژی‌های مبتنی بر هوش مصـنوعی صـورت گرفته اسـت. همان‌طور که قبلاً اشـاره شـد در بین اسـتراتژی‌های تولید دنباله آزمون بهینه برای T=۲ و T=۳ اسـتراتژی SA بهترین عملکرد را دارد. اما زمانی که تعداد مقادیر پارامترها افزایش یابد و یا این‌که مقدار T>۳ باشد نتایج این استراتژی در دسترس نخواهد بود. استراتژی‌های GA و ACA نیز تقریباً شرایط یکسانی با استراتژی SA را دارند و نتایج بهتری را نسـبت به سـایر اسـتراتژی‌ها تولید خواهند کرد. دیگر اسـتراتژی قدرتمند در تولید دنباله آزمون استراتژی HSS است که نتایج قابل‌قبول را تولید می‌کند. استراتژی PSTG در دو مورد قادر به تولید دنباله آزمون نیسـت. الگوریتم پیشـنهادی در این پیکربندی‌ها نتایج برابر با کم‌ترین دنباله آزمون را استخراج نموده است.

آخرین ارزیابی اندازه آرایه بر روی CA (N:t ،۱۰،۲) برای ۲≤T≤۱۰ صـورت خواهد گرفت. در T=۲، T=۳، T=۵ و T=۹ اسـتراتژی HSS بهترین نتایج ممکن را تولید می‌کند و در T=۴ و T=۶ اسـتراتژی‌های HSS، PSTG و راهکار پیشـنهادی دنباله آزمون با کم‌ترین مقدار را اسـتخراج نموده‌اند. اما برای T=۷ راهکار پیشـنهادی بهترین مقدار را تولید می‌کند. در T=۸ دو اسـتراتژی HSS و راهکار پیشـنهادی بهترین مقادیر را تولید می‌کند. در T=۱۰ دو استراتژی مذکور و استراتژی Jenny مطلوب‌ترین نتایج را استخراج نموده‌اند. همان‌طور که در جدول ۶ نشان داده شـده اسـت اسـتراتژی‌های Jenny، Tconfig و IPOG برای T≥۷ قادر به تولید دنباله آزمون نخواهند بود.

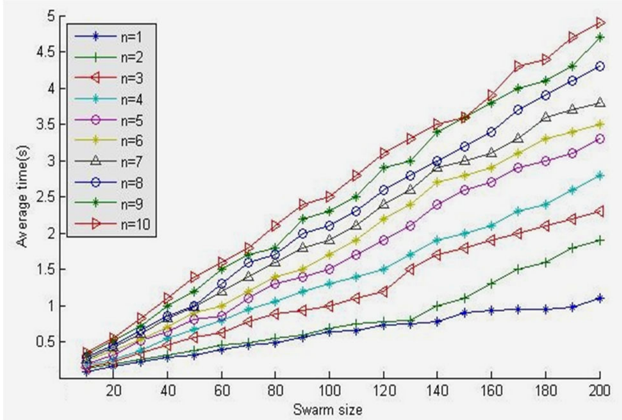
جدول ۶: تعداد نمونه آزمون **CA(N : t،۱۰،۲)**

T	HSS	IPOG	Jenny	TConfig	PSTG	HC-BAT
۲	۷	۱۰	۱۰	۹	۸	۷
۳	۱۶	۱۹	۱۸	۲۰	۱۷	۱۷
۴	۳۷	۴۹	۳۹	۴۵	۳۷	۳۷
۵	۸۱	۱۵۸	۸۷	۹۵	۸۲	۸۵
۶	۱۵۸	۳۵۲	۱۶۹	۱۸۳	۱۵۸	۱۵۸
۷	۲۹۸	NS	۳۱۱	NS	NS	۲۹۴
۸	۴۹۸	NS	۳۲۱	NS	NS	۴۹۸
۹	۵۱۲	NS	۷۸۸	NS	NS	۶۷۳
۱۰	۱۰۲۴	NS	۱۰۲۴	NS	NS	۱۰۲۴

همان‌طوری که در ابتدای این بخش اشـاره شـد ارزیابی "زمان" با توجه به در دسترس نبودن سایر استراتژی‌ها و وابستگی زمان اجرا به سیستم‌عامل و سخت‌افزار، امکان‌پذیر نخواهد بود. اما برای نشان دادن زمان اجرا، راهکار پیشنهادی را بر روی دو پیکربندی (CA(N: ۲، ۶، ۴ و

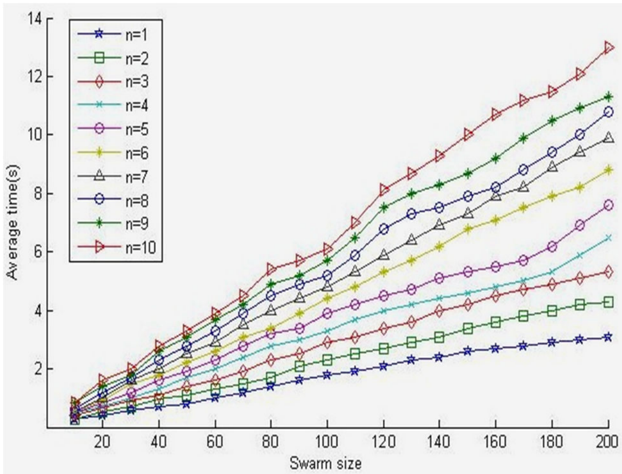
(CA(N: ۳، ۷، ۵ اجرا نموده‌ایم که نتایج آن در شکل ۵ و شکل ۶ آورده شده است. زمان برای جمعیت خفاش‌ها از ۱۰ تا ۲۰۰ و تعداد همسایه‌ها (تپه‌نوردها) از ۱ تا ۱۰ مورد ارزیابی قرار می‌گیرد.

بهترین نتیجه به‌دست‌آمده برای پیکربندی (۴، ۶، CA(N: ۲ تعداد ۲۲ نمونه آزمون است. ا آنجایی‌که این تعداد نمونه آزمون برای جمعیت پایین و تعداد کم همسایه در دسترس نیست زمان برای ۲۵ تکرار در نظر گرفته می‌شود. همان‌طورکه در شکل ۵ مشاهده می‌شود زمان تولید حداکثر ۵ ثانیه به طول می‌انجامد که این مقدار برای استراتژی PSTG با جمعیت به اندازه ۱۷۰ ذره برابر ۷ ثانیه خواهد بود [۱۰].



شکل ۵: میانگین زمان برای پیکربندی (۴، ۶، **CA(N :۲**

در شکل ٦ نیز زمان اجرا بر روی پیکربندی (۵، ۷، CA(N : ۲ محاسبه شده است. بهترین خروجی برای این پیکربندی در این ارزیابی زمان اجرا برای ٤٠ تکرار در نظر گرفته شده است. هنگامی که جمعیت ذرات برابر ۲۰۰ و تعداد تپه‌نوردها ۱۰ باشد زمان اجرا کم‌تر از ۱۳ ثانیه است و زمان اجرا استراتژی PSTG برای همین پیکربندی با جمعیت به تعداد ۱۷۰ ذره به ۱۷ ثانیه خواهد رسید [۱۰].



شکل ۶: میانگین زمان برای پیکربندی (۵،۷،**CA(N :۲**

۶- نتیجه‌گیری و کارهای آتی

در این پژوهش راهکار جدیدی برای تولید خودکار دنباله آزمون در حوزه تست نرم‌افزار پیشنهاد شده است. راهکار پیشنهادی از ترکیب الگوریتم جستجوی تپه‌نوردی و الگوریتم جستجوی خفاش استفاده می‌کند. این راهکار برخلاف سایر الگوریتم‌های فرامکاشفه‌ای با ساختار نه‌چندان پیچیده نتایج به‌مراتب بهتر را استخراج می‌کند. مقدار جدید و سرعت در الگوریتم خفاش با گرایش بسیار زیاد برای بهترین مقدار ایجاد شده، سپس بهترین مقدار به‌عنوان تپه‌نورد به الگوریتم تپه‌نوردی داده می‌شود و همسایه‌های آن ایجاد می‌گردد که در صورت بهبود، با بهترین مقدار جایگزین می‌شود. به‌عنوان یکی دیگر از نقاط قوت این راهکار می‌توان به تابع ایجاد تپه‌نورد جدید اشاره کرد که این تابع با تغییر جزئی بر روی بهترین مقدار باعث دسترسی سریع به دنباله آزمون کمینه شده و نتایج ارزیابی نیز صحت این گفتار را اثبات می‌کند.

راه های متعددی برای گسترش این کار تحقیقاتی وجود دارد. ترکیب سایر الگوریتم‌های فرامکاشفه‌ای ممکن است نتایج بهتر و با سرعت بالاتری را ارائه دهد. اما چالش اصلی برای کار آینده می‌تواند تغییر در تابع برازندگی با شد. همان‌طوری که قبلاً بیان شد در بعضی از ساختارها برای رسیدن به بهترین حالت می‌بایست تعداد جمعیت ذرات را تا حد امکان پایین آورد. به‌وضوح مشخص است که این کاهش جمعیت باعث خواهد شد که در بعضی از تکرارها الگوریتم موفق به یافتن سنگین‌ترین وزن نشود و همین عامل نقطه مثبت برای ادامه الگوریتم خواهد بود. لذا می‌توان نتیجه گرفت که نمونه آزمون با وزن بالا لزوماً بهترین گزینه برای دنباله آزمون نخواهد بود. با این اوصاف تغییر تابع برازندگی می‌تواند تأثیر چشم‌گیری در حوزه تولید خودکار دنباله آزمون داشته باشد.

استراتژی‌های هوش مصنوعی با ساختار پیچیده‌ای که دارند غالباً تا قوه ۳ قادر به تولید دنباله آزمون خواهد بود که با تغییر دادن این الگوریتم‌ها و چشم پوششی از برخی توابع پیچیده، می‌تواند به‌عنوان یک راهکار جدید در کارهای آینده این پژوهش قرار گیرد.

مراجع

[1] N. Tracey, J. Clark, J. McDermid and K. Mander, "A search-based automated test-data generation framework for safety critical systems," in *Systems Engineering for Business Process Change: New Directions*, pp. 174–213, 2002.

[2] P. Arun Babu, C. Senthil Kumar, N. Murali and T. Jayakumar, "An intuitive approach to determine test adequacy in safety-critical software," *ACM SIGSOFT Software Engineering Notes*, vol. 37, no. 5, pp.1-10, 2012.

[3] L. Zhao and W. Luo, "An Algorithm for Reducing Test Suite Based on Interface Parameters," *Computational Intelligence and Software Engineering (CiSE)*, pp. 1-4, 2010.

[۴] زهرا اسلامی مشکنانی، اشکان سامی، «تأثیر اندازه‌های طراحی نسبت به اندازه‌های کد در بهبود کارایی سامانه‌های آزمون خودکار»، مجله مهندسی برق دانشگاه تبریز، جلد ۴۲، شماره ۱، ۱۳۹۲.

[5] L. Luo, *Software Testing Techniques*, Institute for Software Research International, Carnegie Mellon University, Pittsburgh, PA15232, U.S.A., 2001.

[6] S. Nidhra and J. Dondeti, "black box and white box testing techniques–a literature review," *International Journal of Embedded Systems and Applications (IJESA)*, vol. 2, no. 2, pp. 29-50, 2012.

[7] T. El-Ghazali, *Metaheuristics: from design to implementation*, Wiley Publishing, 2009.

[8] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computiong,* Amazon Publishing, 2008.

[9] R. Neapolitan and K. Naimipour, *Foundations of Algorithms Using C++ Pseudocode*, Third Edition, Jones and Bartlett Publishers, 2004.

[10] B. S. Ahmed, K. Z. Zamlia and C. P. Lim, "Application of Particle Swarm Optimization to uniform and variable strengthcovering array construction," *Elsevier: Applied Soft Computing*, vol. 12, no. 4, pp.1330–134, 2012.

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

[12] D. M. Cohen, S. R. Dalal, A. Kajla and G. C. Patton, "The Automatic Efficient Test Generator (AETG) System," *Proceedings of the 5th International Symposium on Software Reliability Engineering*, pp. 303 –309, 1994.

[13] 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.

[14] Y. Lei and K. C. Tai, "In-Parameter-Order: A Test Generation Strategy for Pairwise Testing," in *Proceedings of the 3rd IEEE International Symposium on High-Assurance Systems Engineering*, pp. 254–261, 1998.

[15] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, "IPOG: AGeneral Strategy for T-Way Software Testing," *Proceedings of the 14th Annual IEEE International Conference and Workshops on Engineering of Computer-Based Systems*, pp. 549 –556, 2007.

[16] M.I. Younid and K.Z. Zamli, "MC-MIPOG: A Parallel t-Way Test Generation Strategy for Multicore Systems," *ETRI Journal*, vol. 32, no. 1, pp. 73-82, 2010.

[17] B. Jenkin, *Jenny strategy*, Version 5.0, February-2005, http://burtleburtle.net/bob/math/jenny.html.

[18] M. B. Cohen, C. J. Colbourn, and A. C. H. Ling, "Augmenting Simulated Annealing to Build Interaction Test Suites," *Proceedings of the 14th International Symposium on Software Reliability Engineering*, pp. 394-405, 2003.

[19] X. Chen, Q. Gu, A. Li and D. Chen, "Variable Strength Interaction Testing with an Ant Colony System Approach," *Proceedings of the Asia Pacific Software Engineering Conference*, pp. 160 –167, 2009.

[20] B. S. Ahmed, K. Z. Zamli and C. P. Lim, "Constructing a T-Way Interaction Test Suite Using the Particle Swarm Optimization Approach," *International Journal of Innovative Computing and Information Control*, vol. 8, no. 1, pp. 1–10, 2011.

[21] A. R. A. Alsewari and K. Z. Zamli, "Design and Implementation of a Harmony-Search-Based Variable-Strength t-way Testing Strategy with Constraints Support," *Information Software Technolology*, vol. 54, no. 6, pp. 553–568, 2012.

[22] I. Bashir and, R. A. Paul, "Object-oriented integration testing," *Annals of Software Engineering*, vol. 8, no. 1, pp 187-202, 2001.

[23] R. Kuhn, R. Kacker and Y. Lei, "Practical combinatorial testing–Beyond pairwise testing," *Browse Journals & Magazines*, vol. 10, no. 3, pp. 19-23, 2008.

[24] A. Ganjali, *A Requirements-Based Partition Testing Framework Using Particle Swarm Optimization Technique*, Ph.D. Thesis, Master of Applied Science, In Waterloo, Ontario, Canada, 2008.

[25] S. J. Russell and P. Norvig, *Artificial Intelligence A Modern Approach*, Computer Science, Prentice Hall, 2009.

[26] X. S. Yang, "A New Metaheuristic Bat-Inspired Algorithm," *Nature Inspired Cooperative Strategies for Optimization*, vol. 284, pp. 65-74, 2010.

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

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

[29] ", Int. J. Advance Soft Compu. Appl, vol. 5, no. 3, 2013.

زیرنویس‌ها

۱ T-way testing strategies
۲ coverage array
۳ Non-deterministic Polynomial
۴ Pure computational-based strategies
۵ AI-based strategies
۶ randomness
۷ Orthogonal array testing
۸ local maximum
۹ Plateaux
۱۰ ridges
۱۱ Random-restart hill-climbing
۱۲ Array Size
۱۳ Time
۱۴ Bold
۱۵ Not Available
۱۶ NotSupported