



ORIGINAL RESEARCH ARTICLE

Comparative Analysis of IPOG Strategy Variants for Enhanced Combinatorial T-way Testing Efficiency

Aminu Aminu Muazu^{1,2*}  and Ahmad Sobri Hashim² ¹Computer Science Department, Faculty of Natural and Applied Science, Umaru Musa Yar'adua University, Katsina, Nigeria²Computer & Information Science Department, Faculty of Science and Information Technology, Universiti Teknologi PETRONAS, Perak, Malaysia

ABSTRACT

Enhancing software quality and precision requires thorough testing of various configuration aspects. Combinatorial interaction testing has emerged as a potent method, utilizing strategies like Generalized Input Parameter Order (IPOG), which employs a one-parameter-at-a-time (OPAT) approach. However, the challenge of combinatorial explosion, where the number of inputs grows exponentially with test cases, remains significant. To address this, numerous strategies have been proposed, yet consistently generating optimal tests covering each t-way interaction efficiently remains elusive. This paper introduces a novel variant of the IPOG strategy, termed the enhancing IPOG strategy for uniform interaction testing (eIPOG). We investigate the application of the harmony search algorithm in introducing optimal combinatorial interaction t-way test suites through eIPOG. Evaluating the behavior of such strategies involves assessing their effectiveness in minimizing the test suite size while maximizing coverage. Through experiments on established benchmarking configurations and statistical analysis, we compare eIPOG with other IPOG-based strategies. Both approaches show promise, highlighting the efficacy of the OPAT approach, with eIPOG demonstrating particularly competitive results in nearly every case.

ARTICLE HISTORY

Received May 08, 2024

Accepted July 24, 2024

Published August 01, 2024

KEYWORDS

One-parameter-at-a-time approach, IPOG strategy, combinatorial interaction t-way testing, test case, test suite, harmony search algorithm



© The authors. This is an Open Access article distributed under the terms of the Creative Commons Attribution 4.0 License

(<http://creativecommons.org/licenses/by/4.0/>)

INTRODUCTION

Today, software development follows a modular approach, where small modules are combined to create a complete working system. To ensure the quality of the software, extensive integration testing with numerous test cases is necessary when integrating these modules (Nasser et al., 2016). For software systems that offer extensive customization and configuration options, such as industrial applications, the number of input parameters requiring testing can be substantial. In both scenarios, a significant number of test cases are essential to examine all feasible software interactions within the entire system. The primary aim of a software company is to guarantee that its clients obtain software of excellent quality. Therefore, software should be tested to guarantee its quality and proper operation (Muazu et al., 2024). However, it's important to prioritize software testing to ensure the software meets all functional requirements, preventing issues that customers might notice, avoiding conflicts with involved organizations, and guarding against defects and potentially serious failures (Khaleel & Anan, 2023).

However, inadequate testing may lead to undetected unintended interactions, which could result in the failure

of the entire software system in the future (Muazu et al., 2022). A significant core set of capabilities is served by various system components working together to produce an integrated, highly customizable software system. Any modification to a component's configuration results in an alteration to the system's functionality. As a result, exhaustive testing or evaluating every component to cover every possible scenario is necessary when testing these kinds of software systems (Fadhil et al., 2022). Keep in mind that it might be difficult to thoroughly analyze every option for extremely flexible software configurations because of time, labor, cost, and resource limits. To reduce the test set size into manageable portions, a sampling strategy is therefore required (Chen et al., 2021; Hassan et al., 2020).

Despite being desirable, conducting exhaustive testing of all potential software inputs is impractical due to a common challenge in software testing known as the combinatorial explosion problem, wherein the number of software inputs grows exponentially as the number of test cases increases. This issue could potentially stop test suite generation because of memory limitations. Nowadays, a lot of studies have focused on developing an ideal

Correspondence: Aminu Aminu Muazu. Computer Science Department, Faculty of Natural and Applied Science, Umaru Musa Yar'adua University, Katsina, Nigeria. ✉ aminu.aminu@umyu.edu.ng.

How to cite: Aminu, A. M., & Hashim, A. S. (2024). Comparative Analysis of IPOG Strategy Variants for Enhanced Combinatorial T-way Testing Efficiency. *UMYU Scientifica*, 3(3), 141 – 150. <https://doi.org/10.56919/usci.2433.016>

approach that uses a different technique known as combinatorial interaction testing. Combinatorial interaction testing is employed to identify errors caused when interactions occur between system parameter values (Fadhil et al., 2022). As such, combinatorial interaction testing can save planned expenses and scheduled time while also increasing the efficiency of software testing across several configurations. Yet, producing the most compact t-way test set poses a challenge because of the extensive exploration required, compounded by its classification as a highly complicated (NP-hard (i.e., Non-deterministic Polynomial-time problem)). Thus, new strategies in this area are always welcome.

Additionally, metaheuristic optimization algorithms offer practical solutions within a reasonable time frame to tackle complex challenges in science and engineering (Aminu Muazu et al., 2023). It's important to note that no single metaheuristic holds absolute superiority over its counterparts. This growing interest among researchers and scientists is well-justified. As a result, the metaheuristic search-based method effectively addresses the interaction testing problem (Muazu et al., 2022).

To address the mentioned challenges of reducing testing efforts while maintaining sufficient coverage and software quality, we opted to introduce a novel t-way testing strategy termed improved IPOG strategy (eIPOG). The eIPOG is an OPAT t-way testing strategy that efficiently enhanced the IPOG strategy by combining seeding and constraint supports in a harmony search algorithm (HSA), presenting a promising solution. We characterize the behavior of both eIPOG and other strategies adopting IPOG in terms of test suite size. The results obtained for both strategies indicate the effectiveness of adopting the OPAT approach in combinatorial t-way testing.

According to the works of literature (Aminu Muazu et al., 2022; Ramli et al., 2017), there are three categories of combinatorial t-way testing techniques: uniform interaction strength, variable interaction strength, and input-output-based relations. The uniform interaction strength is a method used in combinatorial t-way testing to combine parameter values based on a consistent level of interaction strength. The term "variable interaction strength" refers to a measure that considers multiple levels of interaction intensity between different variables. The input-output relationship denotes the correlation between a system's inputs and outputs and the combination of parameters that affect a specific output.

Moreover, combinatorial t-way strategies can be divided into three main search-based categories: algebraic-based, computational-based, and metaheuristic-based (Ramli et al., 2017). In the algebraic search-based method, mathematical functions are used for generating test cases with t-way strategies. An example of an algebraic-based strategy is the TConfig (Zamli et al., 2016). In the computational search-based method, the limitations of the algebraic method are overcome. An example of a computational-based strategy is ITTDG (Othman &

Zamli, 2011). Metaheuristic optimization algorithms offer practical solutions within a reasonable time frame to tackle complex challenges in science and engineering. An example of a strategy built on metaheuristic methods is HGHC (Fadhil et al., 2023).

Additionally, when creating a test case, t-way techniques are divided into two main methods: the one-test-at-a-time (OTAT) approach and the one-parameter-at-a-time (OPAT) approach (Alsariera & Zamli, 2015; Ramli et al., 2017). The OTAT method adds one test case at a time to build the test suite, while the OPAT approach starts with a full test suite and adds parameters one by one, possibly including more test cases for full interaction coverage. Moreover, some combinations of sampled test data may be undesirable or impractical, so it's important to follow support restrictions. Similarly, there might be a specific set of combinations that must be used. However, the established t-way combinatorial strategies that adopt the original OPAT (IPOG) approach are described in the following paragraphs.

The IPOG strategy (Lei et al., 2007) generalized the original OPAT approach IPO by a combination of all horizontal algorithms and the vertical algorithm in the generation of the t-way test case. In their endeavor, they outlined a testing algorithm called FireEye, alternatively referred to as IPOG-D-Test, designed to aid in generating t-way combinations. Quickly, a variant has been applied to improve the execution time of IPOG known as IPOG-D (Lei et al., 2008). Two variants of IPOG are proposed in (Wang Ziyuan et al., 2007), known as ParaOrder and ReqOrder strategies. Both of them support Input-Output Relation. These approaches were suggested after analyzing a flaw in the union algorithm, where the test suite size consistently remains large due to the inclusion of 'don't care' values. However, these values are not relevant to the current coverage requirements.

MIPOG strategy is proposed in (Younis et al., 2008), which systematically evaluates all input parameter values to prioritize combinations with the highest number of uncovered tuples. Additionally, it reorganizes sets of t-way combinations in decreasing order of size, selecting the initial tuple for further combination with other valid tuples. G_MIPOG strategy proposed in (Younis et al., 2008) utilizes a grid system to create t-way test sets. The goal is to enhance and adapt the MIPOG strategy to function effectively within a grid framework. MC-MIPOG is proposed in (Younis & Zamli, 2010) as a parallel approach tailored for t-way testing on multi-core architectures. In contrast to the original MIPOG model, it pioneers a new method of leveraging multicore systems by minimizing control and data dependencies, ensuring effective utilization of multicore setups. TS_OP strategy is proposed by (Soh et al., 2013) as a distributed approach for t-way testing, employing Map and Reduce procedures on a network of workstations and utilizing Tuple Space Awareness for its implementation.

ACTS strategy is proposed by (Yu et al., 2013), who used the modified versions of the original IPOG to strike a trade-off between the size of the test suite and the time it takes to execute. OPAT-HS strategy is introduced in (Alsewari et al., 2018), which adopts the original IPOG inside HSA with lower interaction strength. PwiseHA (Aminu Muazu & Maiwada, 2020) utilizes the same mechanism as the OPAT-HS strategy but with a higher level of interaction strength than the OPAT-HS. Recently, SCIPOG was introduced by (Aminu Muazu et al., 2023) as a computational base that modifies the original IPOG to handle seeding and constraints in improving software quality. However, other strategies like GAMIPOG (Younis, 2020) and SCIPOG-VS (Muazu et al., 2023) support variable interaction strength in the OPAT approach. The GAMIPOG integrates elements from both the Genetic Algorithm and with MIPOG strategy to produce an optimal test suite, while SCIPOG-VS is a computational base that adopts the OPAT approach to support seeding and constraint.

As highlighted in the above paragraphs, significant progress has been made in the generation of combinatorial interaction t-way tests by adopting the original IPOG strategy in different modifications. Consequently, there is a recognized need, as recently indicated in (Alazzawi et al., 2022; Fadhil et al., 2022) surveys, to explore new strategies for generating test suites, aiming for increased effectiveness and optimality in t-way testing. As a result, the current paper will concurrently integrate seeding and constraint support within the original IPOG strategy and HSA for uniform t-way testing for characterization with other variants of IPOG strategies.

METHODOLOGY

The IPOG strategy and HSA are essential for achieving the most efficient test suite sizes in eIPOG. Understanding the eIPOG strategy involves exploring how IPOG and HSA work. As discussed in Section, the IPOG strategy is generalized from an existing strategy known as IPO. Their primary effort in generalization was accommodating the combinatorial growth in parameter value combinations (Lei et al., 2007). This approach aims to achieve an optimal test size and faster execution time. The IPOG implementation accommodates interaction strengths up to 6 ($t \leq 6$). Figure 1 depicts the core concept of the IPOG strategy algorithm.

HSA is a modern optimization technique inspired by natural processes, particularly the way musicians create music (Geem et al., 2001). Musicians draw from a repertoire of music pitches stored in memory, experimenting with combinations much like HSA refines its solutions over iterations, using past discoveries to generate new ones. HSA operates with three key rules: harmony memory consideration rate, pitch adjustment

rate, and random selection. Figure 2 illustrates the essence of HSA.

The proposed strategy, eIPOG, is rooted in the original IPOG strategy mechanism, which comprises the seeding, constraint, and combination algorithms. A detailed concept of the operation of the eIPOG strategy is provided in Figure 3. The seeding algorithm incorporates user-specified test data straight into the final test suite, while the constraint algorithm iteratively selects test cases that meet specified constraints. The combination algorithm facilitates parameter interaction using the IPOG mechanism within HSA for uniform interaction testing.

Moreover, the combination algorithm serves as the primary method for generating the optimal test suite following the IPOG strategy within HSA. For instance, in a system with t or more parameters, our eIPOG strategy constructs a t -way test set for the initial t parameters. It then extends this set to include the $t+1$ parameters and continues until a t -way test set is created for all system parameters. Similar to IPOG, eIPOG employs horizontal and vertical growth. However, eIPOG's approach to horizontal and vertical growth differs from IPOG's in that it integrates seeding and constraint considerations within the HSA to optimize the number of generated tests, ensuring each t -way interaction is covered by a single test.

In the context of experimental setup, the eIPOG strategy operates on an HP laptop equipped with Windows 10, 8GB RAM, with an Intel Core i5. Experimental findings are compared with results from other strategies in publications and displayed in tabular form. In cases where specific configuration results are unavailable, they are marked as "NA" (not available), and if the strategy does not endorse a particular interaction strength configuration, it is denoted as "NS" (not supported). Blueish cells highlight the best (smallest) sizes. Notably, eIPOG supports a maximum interaction strength of 5 ($t \leq 5$).

Furthermore, regarding the settings for HSA in our eIPOG strategy, such as improvisation, Harmony Memory Size (HMS), Harmony Memory Consideration Rate (HMCR), and Pitch Adjustment Rate (PAR), we need to consider some factors. As mentioned in (Alsewari et al., 2020), using a high value for improvisation may not always be helpful if there's no improvement in the solution. Conversely, a low value might not give good results, especially during transitions between iterations. Also, having a very large HMS could slow down computation, while a small one may decrease the chance of finding a good solution. Therefore, we choose to have 10 improvisations and an HMS of 10. We also keep HMCR at 0.90 (with a 90% probability) and PAR at 0.2 (with a 20% probability), following the recommendation of (Geem et al., 2001) to get the best results.

```

Algorithm IPOG-Test (int t, parameterSet ps)
{
  1. Initialize test set ts to be an empty set
  2. Denote the parameters in ps, in an arbitrary order, as P1, P2, ..., and Pn
  3. Add into test set ts a test for each combination of values of the first t parameters
  4. For(int i=t+1; i≤n; i++) {
  5. Let π be the set of t-way combinations of values involving parameter Pi and t-1
     parameter among the first i-1 parameters
  6. //horizontal extension for parameter Pi
  7. For(each test t=(v1,v2,...,vt-1) in test set ts) {
  8.   Choose a value Vi of Pi and replace t with t'=(V1,V2,...,Vi-1,Vi) so that t'
     covers the most number of combinations of values covered by t
  9.   Remove from π the combinations of values covered by t'
  10. }
  11. // vertical extension for parameter Pi
  12. For(each combination α in set π) {
  13.   If(there exist a test that already covers α) {
  14.     Remove α from π
  15.   } else {
  16.     Change an existing test, if possible, or otherwise add a new test to cover
     α and remove it from π
  17.   }
  18. }
  19. }
  20. Return ts
}

```

Figure 1: The IPOG strategy algorithm (Lei et al., 2007)

```

ALGORITHM: Harmony Search Algorithm
  1. Harmony search parameters settings:
  2.   Define the objective function f(x).
  3.   Define HMCR.
  4.   Define PAR.
  5.   Define HMS.
STEP_1: 6. HM initialization
  7.   Generate harmonies in HM randomly.
STEP_2: 8. Improvise a new harmony.
  9.   for (t=0 to MAX_NUMIteration)
 10.     for (d=1 to NUMBER_OFDiminsion)
 11.       if (HMCRRandomValue ≤< HMCR), then //improvise
                                     //locally
 12.         Chose a value from HM for variable d.
 13.         if (PARRandom ≤< PAR), then
 14.           Adjust the value.
 15.         else
 16.           Do not Adjust.
 17.         else
 18.           Chose a random value. //improvise globally
 19.       end loop
STEP_3: 20. Update the HM.
 21.   Accept the new harmony and add to HM if better than
     the worst.
STEP_4: 22. Check the stopping condition.
 23.   end loop
 24. RETURN BEST HARMONY

```

Figure 2: The harmony search algorithm (Geem et al., 2001)

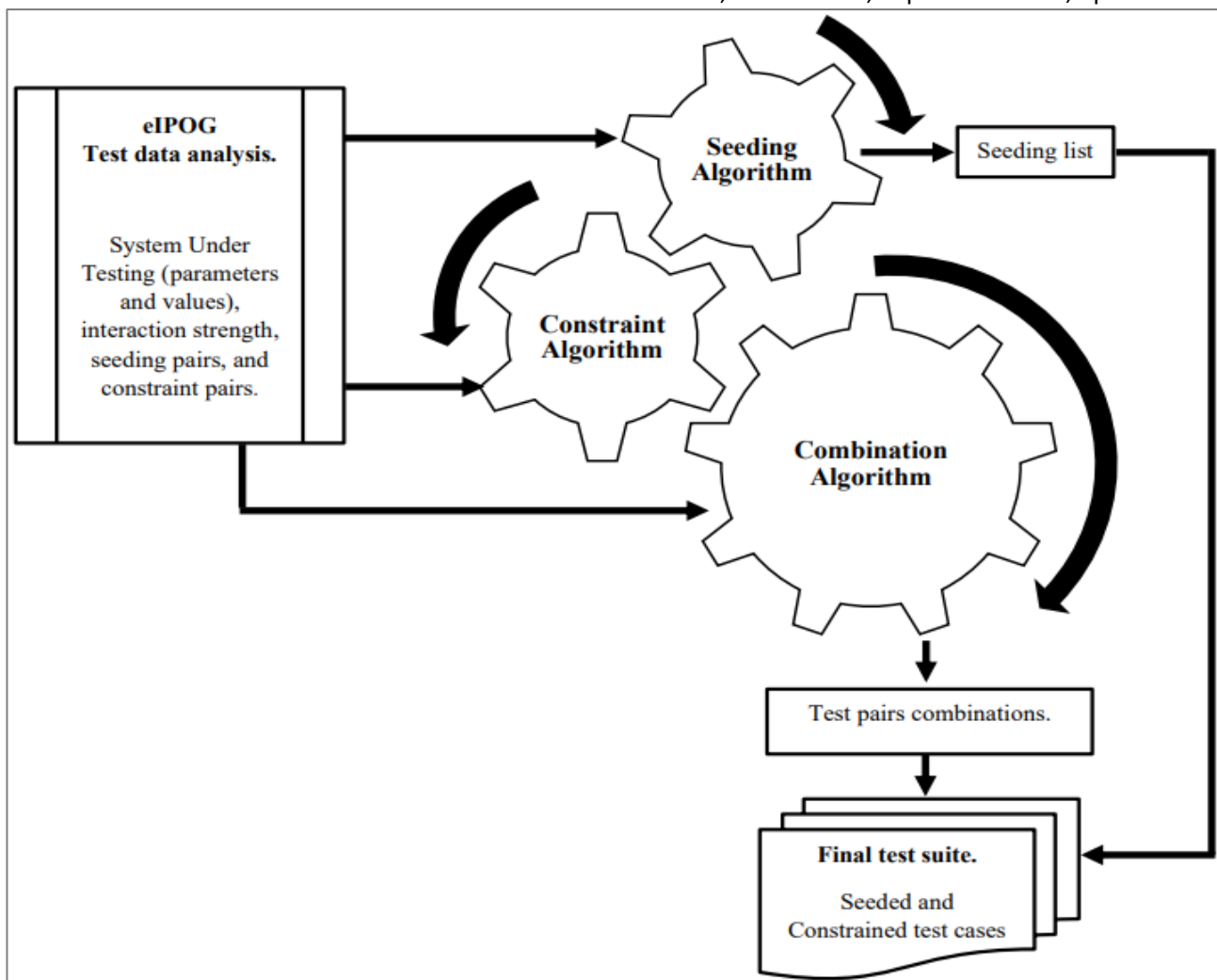


Figure 3: The concept of operation for eIPOG strategy.

RESULTS AND DISCUSSION

The evaluation aims to examine how eIPOG compares to IPOG and its variants in terms of minimizing test suite size while maximizing coverage. To achieve this, we implement the eIPOG strategy on three established benchmarking configurations as outlined in (Lei et al., 2007) and compare the results with those obtained using IPOG and other IPOG variants. In this setup, we've outlined a total of three sets of experiments, as detailed below:

- **Experiment set 1** = CA (N; t, 5¹⁰), the benchmarking involves test configurations where the number of parameters and values remains fixed at 10 and 5, respectively. However, the interaction strength ranges from 2 to 5. The outcomes are displayed in Table 1.
- **Experiment set 2** = CA (N; 4, 5^p), the benchmarking involves test configurations where the interaction strength and values remain fixed at 4 and 5, respectively. However, the number of parameters ranges from 5 to 15. The results are displayed in Table 2.
- **Experiment set 3** = CA (N; 4, v¹⁰), the benchmarking assesses test configurations with a

consistent number of parameters as 10 and interaction strength as 4, while the values vary from 2 to 10. The results are shown in Table 3.

Referring to the experiment set 1 result in Table 1, most of the strategies do not support the configuration, as they are recorded as “NS”. In some instances, they show “NA” in the strategy's results. However, the remaining strategies—namely, pioneer IPOG, MIPOG, MC-MIPOG, SCIPOG, and our eIPOG—have all their results recorded. When the interaction strength is 2, OPAT-HS emerges as the best-performing strategy. Conversely, when the interaction strength ranges from 3 to 5, our eIPOG outperforms IPOG and its variants (MIPOG, MC-MIPOG, and SCIPOG) in all cases. Similarly to the results in Figure 4, however, only five strategies— IPOG, MIPOG, MC-MIPOG, SCIPOG, and our eIPOG—are displayed, as they have complete results.

The next evaluation in this paper pertains to experiment set 2, as discussed earlier. This evaluation examines the effect of an increasing number of parameters on time in the strategies listed in Table 2. For this evaluation, the results of IPOG, MIPOG, MC-MIPOG, and SCIPOG are available, providing insights into the behavior of their successor, the proposed eIPOG strategy. Additionally,

Table 2 highlights eIPOG, MIPOG, MC-MIPOG, and SCIPOG as the strongest strategies in terms of performance, all exhibiting similar competitiveness, while IPOG ranks the lowest. The statistical results of this evaluation are depicted in Figure 5, which mirrors the findings in Table 2. It's evident from both sets of results that eIPOG performs well when the number of parameters ranges from 7 to 13. However, for parameter counts 5, 6, 14, and 15, MIPOG and MC-MIPOG outperform eIPOG. This suggests that eIPOG struggles to yield favorable results when the parameter count is below 5 or exceeds 13.

The final evaluation, as displayed in Table 3, focuses on experiment set 3. This characterization is extensive, accommodating parameters with up to 10 values. Consequently, we assessed the proposed eIPOG strategy alongside IPOG and its variants. The results in Table 3 demonstrate that for large parameter values ranging from 2 to 10, the eIPOG strategy outperformed all other strategies, including the pioneer IPOG, in terms of test suite size, as also illustrated in Figure 6. Through meticulous characterization of IPOG and its variants, it can be inferred that for values exceeding 10, our proposed eIPOG strategy may even surpass other strategies.

Because of the intricate design and frequent repetition of the OPAT approach of generating combinatorial t-way test cases, metaheuristic-based t-way strategies like

OPAT-HS, PwiseHA, GAMIPOG, and our eIPOG consistently produce outstanding results when creating test suites. Similarly, computational strategies such as IPOG, ParaOrder, ReqOrder, TS_OP, G-MIPOG, MIPOG, MC-MIPOG, ACTS, SCIPOG, and SCIPOG_VS. However, due to unavailable results in the respective benchmark configurations marked as NA or NS, OPAT-HS, PwiseHA, ParaOrder, ReqOrder, TS_OP, G-MIPOG, GAMIPOG, SCIPOG, and SCIPOG_VS were not included in the characterization evaluation. Nonetheless, they have demonstrated commendable performance in their respective publications. Among the evaluated strategies, eIPOG surpasses computational-based strategies like MIPOG, MC-MIPOG, SCIPOG, and the original IPOG.

Now, let's explore the rationale behind the OPAT approach for generating combinatorial interaction t-way test cases. Managing the risk of faults following interactions between input parameters presents a challenge when devising a practical test plan. The OPAT approach provides a viable solution by dividing each domain into segments and choosing typical values from each segment to create test cases. As a result, this approach reduces the number of test cases needed. However, it assumes that balancing the risk of interaction among non-representative parameters while completing system testing using representative values is feasible within a reasonable budget.

Table 1: Characterizing eIPOG against IPOG and other IPOG variants for experiment set 1.

t-way strength	2	3	4	5
IPOG	48	308	1,843	10,119
ParaOrder	NS	NS	NS	NS
ReqOrder	NS	NS	NS	NS
MIPOG	45	281	1,643	8,169
G-MIPOG	NA	NA	NA	NA
MC-MIPOG	45	281	1,643	8,169
TS_OP	NA	NA	NA	NA
ACTS	NA	NA	NA	NA
OPAT-HS	43	NA	NS	NS
PwiseHA	NA	NA	NA	NS
SCIPOG	75	369	1,709	7,781
GAMIPOG	NS	NS	NS	NS
SCIPOG-VS	NS	NS	NS	NS
eIPOG	46	278	1,492	7,522

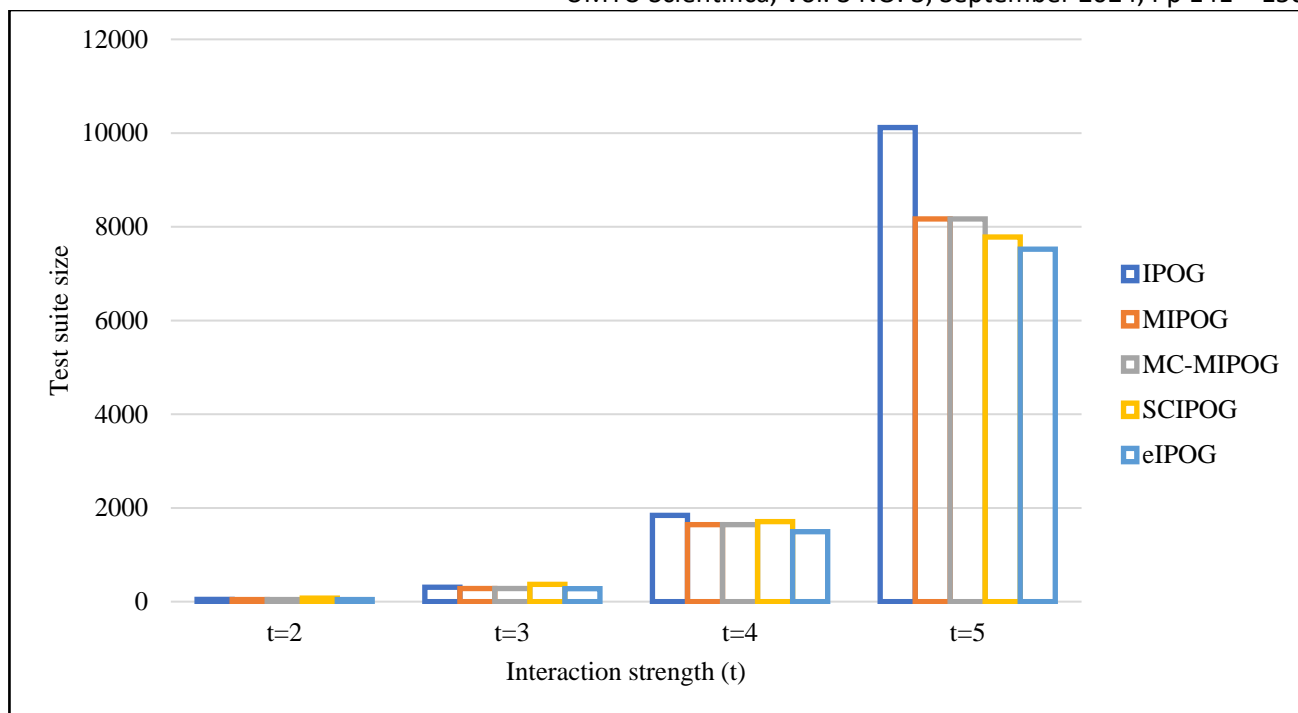


Figure 4: Illustration of test suite size difference between eIPOG, MIPOG, MC-MIPOG, SCIPOG, and IPOG for experiment set 1.

Table 2: Characterizing eIPOG against IPOG and other IPOG variants for experiment set 2.

No of parameter	5	6	7	8	9	10	11	12	13	14	15
IPOG	784	1,064	1,290	1,491	1,677	1,843	1,990	2,132	2,254	2,378	2,497
ParaOrder	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS
ReqOrder	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS
MIPOG	625	625	1,125	1,384	1,543	1,643	1,722	1,837	1,956	2,051	2,150
G-MIPOG	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
MC-MIPOG	625	625	1,125	1,384	1,543	1,643	1,722	1,837	1,956	2,051	2,150
TS_OP	NA	NA	NA	NA	NA	1,777	NA	NA	NA	NA	NA
ACTS	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
OPAT-HS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS
PWiseHA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
SCIPOG	815	992	1,179	1,361	1,530	1,709	1,892	2,068	2,214	2,443	2,647
GAMIPOG	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS
SCIPOG-VS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS
eIPOG	715	854	1,008	1,129	1,361	1,488	1,662	1,773	1,891	2,068	2,212

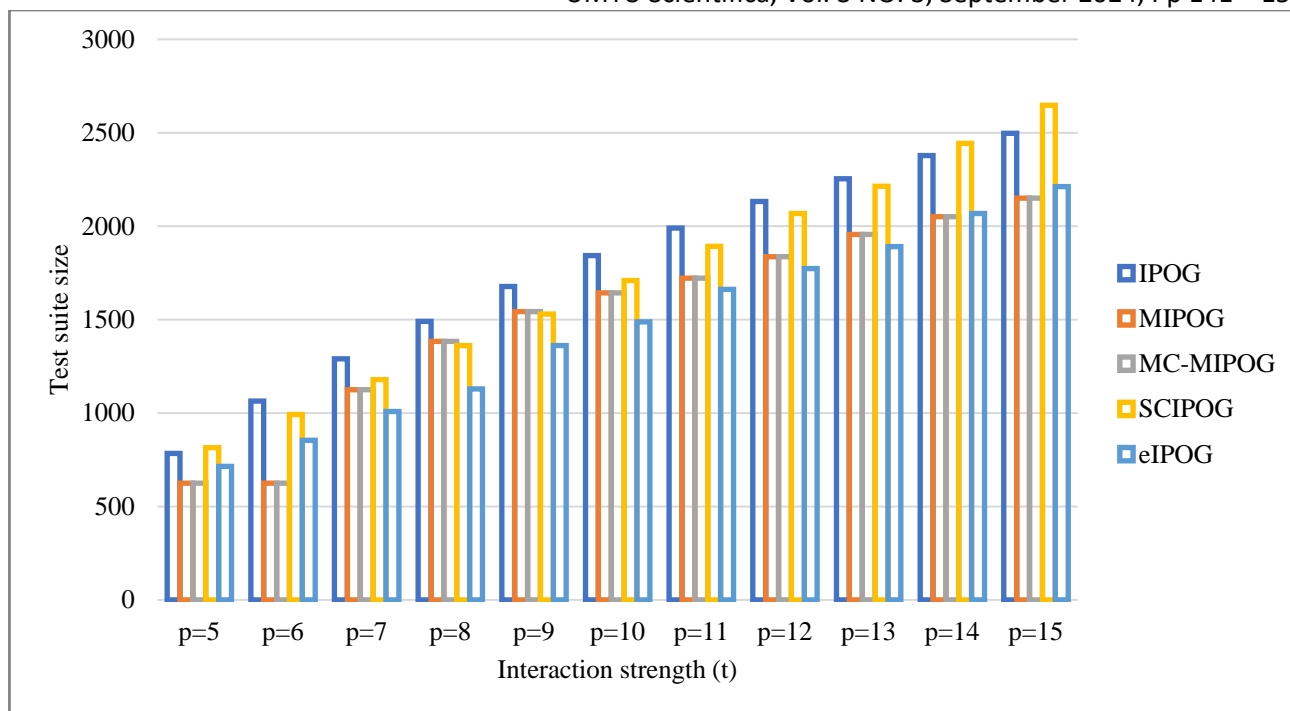


Figure 5: Illustration of test suite size difference between eIPOG, MIPOG, MC-MIPOG, SCIPOG, and IPOG for experiment set 2.

Table 3: Characterizing eIPOG against IPOG and other IPOG variants for experiment set 3.

No of values	2	3	4	5	6	7	8	9	10
IPOG	46	229	649	1,843	3,808	7,061	11,993	19,098	28,985
ParaOrder	NS	NS	NS	NS	NS	NS	NS	NS	NS
ReqOrder	NS	NS	NS	NS	NS	NS	NS	NS	NS
MIPOG	43	217	637	1,643	3,657	5,927	11,355	18,036	27,306
G-MIPOG	NA	NA	NA	NA	NA	NA	NA	NA	NA
MC-MIPOG	43	217	637	1,643	3,657	5,927	11,355	18,036	27,306
TS_OP	NA	NA	NA	NA	NA	NA	NA	NA	NA
ACTS	NA	NA	NA	NA	NA	NA	NA	NA	NA
OPAT-HS	NS	NS	NS	NS	NS	NS	NS	NS	NS
PWiseHA	NA	NA	NA	NA	NA	NA	NA	NA	NA
SCIPOG	30	191	672	1,709	3,689	6,909	11,969	19,419	29,905
GAMIPOG	NS	NS	NS	NS	NS	NS	NS	NS	NS
SCIPOG-VS	NS	NS	NS	NS	NS	NS	NS	NS	NS
eIPOG	29	189	595	1,508	3,331	5,812	9,776	17,990	27,011

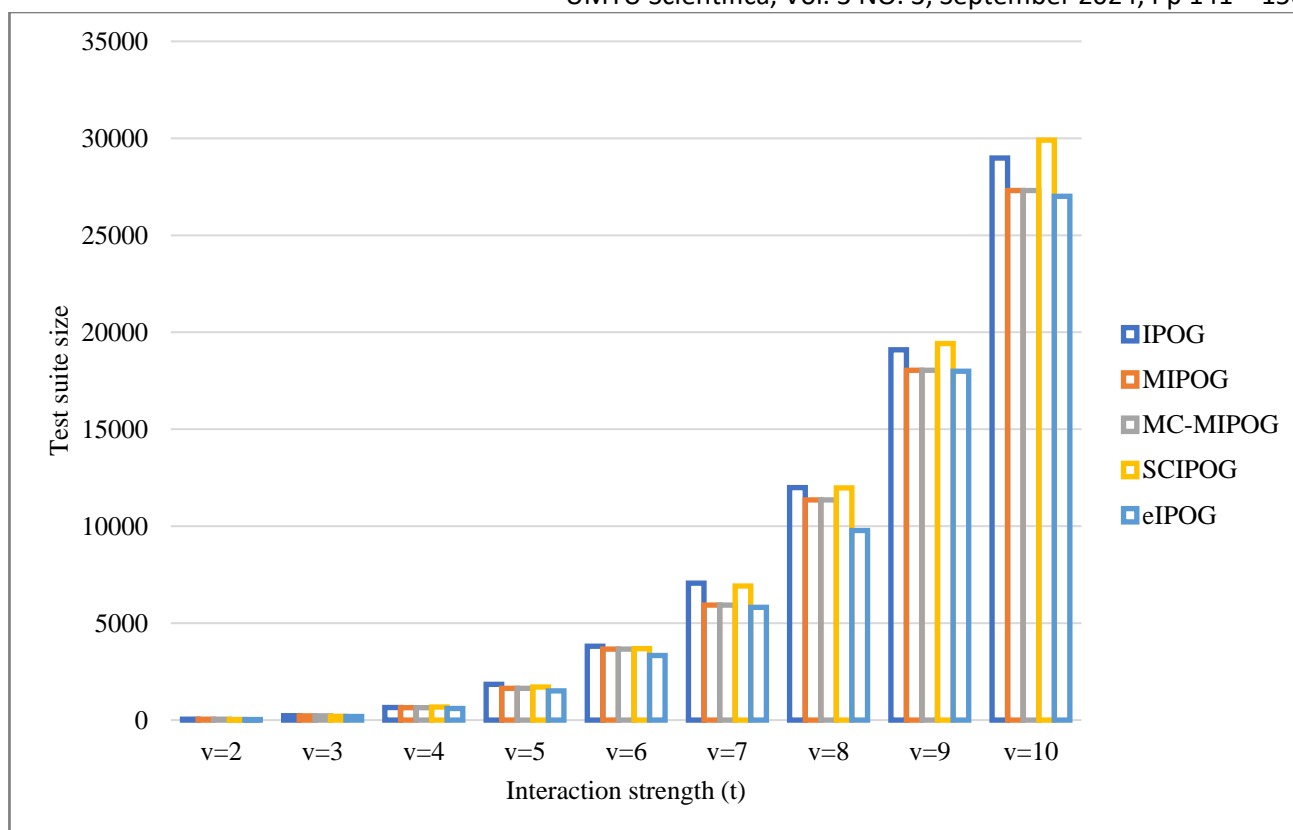


Figure 6: Illustration of test suite size difference between eIPOG, MIPOG, MC-MIPOG, SCIPOG, and IPOG for experiment set 3.

CONCLUSION

In this paper, we introduce eIPOG, an enhanced version of the IPOG strategy that addresses uniform interaction strength, seeding, and constraints using the harmony search algorithm. By modifying the bit structure of test case generation in eIPOG, we have significantly improved IPOG's performance. We evaluate the behavior of eIPOG alongside the original IPOG and its variations, assessing their efficacy in reducing test suite size while maximizing coverage using established benchmark configurations. Experimental results show that eIPOG performs competitively against most existing strategies, including MIPOG, followed by MC-MIPOG, SCIPOG, and the original IPOG. In our future research, we aim to explore other metaheuristic approaches to integrate them with eIPOG for enhanced efficiency.

ACKNOWLEDGEMENT

The authors express gratitude for the support of this research provided by Universiti Teknologi Petronas Malaysia for the utilization of their resources. Additionally, we are deeply thankful to the anonymous reviewers for their insightful comments and suggestions, which significantly enhanced the quality of this paper.

REFERENCES

Alazzawi, A. K., Rais, H. M., Basri, S., Alsariera, Y. A., Imam, A. A., Abed, S. A., Balogun, A. O., & Kumar, G. (2022). Recent t-way Test Generation

Strategies Based on Optimization Algorithms: An Orchestrated Survey. *International Conference on Artificial Intelligence for Smart Community: AISC 2020, 17–18 December, Universiti Teknologi Petronas, Malaysia*, 1055–1060.

Alsariera, Y. A., & Zamli, K. Z. (2015). A Bat-inspired strategy for t-way interaction testing. *Advanced Science Letters*, 21(7). [\[Crossref\]](#)

Alsewari, A., Mu'aza, A. A., Rassem, T. H., Tairan, N. M., Shah, H., & Zamli, K. Z. (2018). One-Parameter-at-a-Time Combinatorial Testing Strategy Based on Harmony Search Algorithm OPAT-HS. *Advanced Science Letters*, 24(10). [\[Crossref\]](#)

Alsewari, A. R. A., Poston, R., Zamli, K. Z., Balfaqih, M., & Aloufi, K. S. (2020). Combinatorial test list generation based on Harmony Search Algorithm. *Journal of Ambient Intelligence and Humanized Computing*. [\[Crossref\]](#)

Aminu Muazu, A., & Maiwada, U. D. (2020). PwiseHA: Application of Harmony Search Algorithm for Test Suites Generation using Pairwise Techniques. *International Journal of Computer and Information Technology*, (2279–0764), 9(4). [\[Crossref\]](#)

Aminu Muazu, A., Sobri Hashim, A., & Sarlan, A. (2022). Application and Adjustment of “don't care” Values in t-way Testing Techniques for Generating an Optimal Test Suite. *Journal of Advances in Information Technology*, 13(4), 347–357. [\[Crossref\]](#)

- Aminu Muazu, A., Sobri Hashim, A., Sarlan, A., & Abdullahi, M. (2023). SCIOG: Seeding and constraint support in IPOG strategy for combinatorial t-way testing to generate optimum test cases. *Journal of King Saud University - Computer and Information Sciences*, 35(1), 185–201. [\[Crossref\]](#)
- Chen, J., Chen, J., Cai, S., Chen, H., Zhang, C., & Huang, C. (2021). A Test Case Generation Method of Combinatorial Testing based on t-way Testing with Adaptive Random Testing. *Proceedings - 2021 IEEE International Symposium on Software Reliability Engineering Workshops, ISSREW 2021*. [\[Crossref\]](#)
- Fadhil, H. M., Abdullah, M. N., & Younis, M. I. (2022). Combinatorial Testing Approaches: A Systematic Review. *IRAQI JOURNAL OF COMPUTERS, COMMUNICATIONS, CONTROL AND SYSTEMS ENGINEERING*, 22(4), 60–79. [\[Crossref\]](#)
- Fadhil, H. M., Abdullah, M. N., & Younis, M. I. (2023). Innovations in t-way test creation based on a hybrid hill climbing-greedy algorithm. *LAES International Journal of Artificial Intelligence*, 12(2), 794–805. [\[Crossref\]](#)
- Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A New Heuristic Optimization Algorithm: Harmony Search. *Simulation*, 76(2). [\[Crossref\]](#)
- Hassan, A. A., Abdullah, S., Zamli, K. Z., & Razali, R. (2020). Combinatorial test suites generation strategy utilizing the whale optimization algorithm. *IEEE Access*, 8. [\[Crossref\]](#)
- Khaleel, S. I., & Anan, R. (2023). A review paper: optimal test cases for regression testing using artificial intelligent techniques. *International Journal of Electrical and Computer Engineering*, 13(2). [\[Crossref\]](#)
- Lei, Y., Kacker, R., Kuhn, D. R., Okun, V., & Lawrence, J. (2007). IPOG: A General Strategy for T-Way Software Testing. *14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07)*.
- Lei, Y., Kacker, R., Kuhn, D. R., Okun, V., & Lawrence, J. (2008). IPOG-IPOG-D: Efficient test generation for multi-way combinatorial testing. *Software Testing Verification and Reliability*, 18(3), 125–148. [\[Crossref\]](#)
- Muazu, A. A., Hashim, A. S., Maiwada, U. D., Isma'ila, U. A., Yakubu, M. M., & Ibrahim, M. A. (2024). Pairwise test case generation with harmony search, one-parameter-at-at-time, seeding, and constraint mechanism integration. *International Journal of Electrical and Computer Engineering*, 14(3), 3137–3149. [\[Crossref\]](#)
- Muazu, A. A., Hashim, A. S., Maiwada, U. D., & Muppidi, A. (2023). Enhanced Version of Seeding and Constraint support in IPOG strategy for Variable Strength Interaction T-way Testing. *Malaysian Journal of Computer Science*, 36(4), 381–403. [\[Crossref\]](#)
- Muazu, A. A., Hashim, A. S., & Sarlan, A. (2022). Review of Nature Inspired Metaheuristic Algorithm Selection for Combinatorial t-Way Testing. *IEEE Access*, 10, 27404–27431. [\[Crossref\]](#)
- Muazu, A. A., Hashim, A. S., Sarlan, A., & Maiwada, U. D. (2022). Proposed Method of Seeding and Constraint in One-Parameter-At-a-Time Approach for t-way Testing. *2022 International Conference on Digital Transformation and Intelligence (ICDI)*, 39–45. [\[Crossref\]](#)
- Nasser, A. B., Alsewari, A. A., Aminu Muazu, A., & Zamli, K. Z. (2016). Comparative Performance Analysis of Flower Pollination Algorithm and Harmony Search based strategies: A Case Study of Applying Interaction Testing in the Real World. *2nd International Conference on New Directions in Multidisciplinary Research & Practice (NDMRP)*, 3, 51–51. globalilluminators.org
- Othman, R. R., & Zamli, K. Z. (2011). ITTDG: Integrated T-way test data generation strategy for interaction testing. *Scientific Research and Essays*, 6(17), 3638–3648. [\[Crossref\]](#)
- Ramli, N., Othman, R. R., Abdul Khalib, Z. I., & Jusoh, M. (2017). A Review on Recent T-way Combinatorial Testing Strategy. *MATEC Web of Conferences*, 140. [\[Crossref\]](#)
- Soh, Z. H. C., Abdullah, S. A. C., & Zamli, K. Z. (2013). A Distributed T-Way Test Suite Generation Using “One-Parameter-at-a-Time” Approach. In *Int. J. Advance Soft Comp. Appl* (Vol. 5, Issue 3).
- Wang Ziyuan, Nie Changhai, & Xu Baowen. (2007). Generating Combinatorial Test Suite for Interaction Relationship. *Fourth International Workshop on Software Quality Assurance : In Conjunction with the 6th ESEC/FSE Joint Meeting*, 3(4), 115.
- Younis, M. (2020). GAMIPOG: A Deterministic Genetic Multi-Parameter-Order Strategy for the Generation of Variable Strength Covering Arrays. In *Journal of Engineering Science and Technology* (Vol. 15, Issue 5). researchgate.net
- Younis, M. I., & Zamli, K. Z. (2010). MC-MIPOG: A parallel t-way test generation strategy for multicore systems. *ETRI Journal*, 32(1), 73–83. [\[Crossref\]](#)
- Younis, M. I., Zamli, K. Z., & Ashidi Mat Isa, N. (2008). MIPOG-Modification of the IPOG Strategy for T-Way Software Testing. researchgate.net
- Younis, M. I., Zamli, K. Z., & Isa, N. A. M. (2008). A strategy for grid based T-Way test data generation. *1st International Conference on Distributed Frameworks and Application, DFmA 2008*, 73–78. [\[Crossref\]](#)
- Yu, L., Lei, Y., Kacker, R. N., & Kuhn, D. R. (2013). ACTS: A combinatorial test generation tool. *Proceedings - IEEE 6th International Conference on Software Testing, Verification and Validation, ICST 2013*, 370–375. [\[Crossref\]](#)
- Zamli, K. Z., Alkazemi, B. Y., & Kendall, G. (2016). A Tabu Search hyper-heuristic strategy for t-way test suite generation. *Applied Soft Computing Journal*, 44, 57–74. [\[Crossref\]](#)