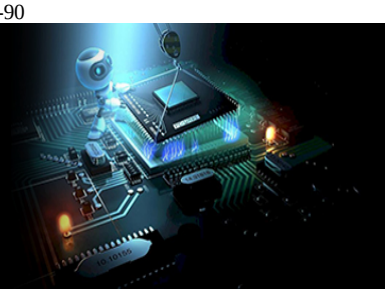
International Journal of Engineering in Computer Science



E-ISSN: 2663-3590
P-ISSN: 2663-3582
www.computersciencejournals.com/ijecs
IJECS 2025; 7(1): 85-90
Received: 05-01-2025
Accepted: 07-02-2025

Abhishek Pandey
Department of Computer Science and Engineering, Chandigarh University, Punjab, India

Corresponding Author:
Abhishek Pandey
Department of Computer Science and Engineering, Chandigarh University, Punjab, India

Test case optimization using chaos based heuristics

Abhishek Pandey

DOI: https://www.doi.org/10.33545/26633582.2025.v7.i1b.159

Abstract
Software testing is a significant component of software verification and validation paradigm. Software testing ensures the confidence in the software. Software testing also provides various quality parameters necessary for deployment of the software. It is no surprise that software testing requires almost 50% of time and cost of the software project. In this paper various chaotic evolutionary algorithm is used for test case optimization in software testing

Keywords: Software testing, chaotic algorithm, evolutionary algorithm, test case optimization, validation

Introduction

Software engineering is an application of systematic, quantifiable and disciplined approach towards development, operation and maintenance of software and at the same time, it must accomplish user requirements and desires. Almost half of the time and cost spent on software projects is due to software testing activities. Software testing is a significant component of software verification and validation paradigm. Software testing ensures the confidence in the software. Software testing also provides various quality parameters necessary for deployment of the software. It is no surprise that software testing requires almost 50% of time and cost of the software project. Studies also revealed that software testing is responsible for nearly fifty per cent of total cost of the software projects. Hence, Software testing requires a lot of human efforts to find out the quality test case and processes for quality enhancement.

Software testing performs an important task to find an error in the program. Testing requires almost more than half of the total cost and time invested in a software project (Myers, Sandler, & Badgett, 2011, 2015) [3, 4]. Software testing adds value to the software program that in turn is responsible for improving software quality. Various efforts for automating testing tasks are present in the literature (Korel, 1983, 1990, 1996) [5, 6, 7]. Different approaches to software testing such as symbolic execution and Concolic testing exists in the software engineering literature (King, 1976; Cadar & Sen, 2013; Sen, Marinov, &Agha, 2011) [9, 10]. Software engineering problems often require solutions to satisfy various conflicting objectives such as finding the minimal set of test cases that achieve maximum branch coverage (Harman &Jones, 2001). Harman *et al.* (2003) [12] reformulated software engineering problem as a search problem and attacked with optimization algorithms. Search based software engineering is a term coined by authors of Harman and Jones (2001) [12] advocates the new field of software engineering that reformulates the software engineering problem as a search problem. Test case quality assessment and optimization is trivial for all software projects (Pandey & Banerjee, 2018) [18]. Various metaheuristics applied for software testing problems ensures generation of quality test data. Recently it has been shown that introduction of chaos improves the performance of metaheuristic algorithms (Liu *et al.*, 2015; Afrabandpey, 2014) [14, 15].

Software testing requires being automated and optimized to reduce time and cost of software development. Search-Based Software Engineering (SBSE) is a field that is more concerned of applications of meta-heuristic search algorithms in software testing problems. This is to reformulate the software engineering task as an optimization problem. This will require an initial representation of the individuals comprising the solutions and a fitness function that capture the unique properties of the system.

In software-testing generating valid input test-data refers to test data generation problem.

Many tools and techniques exists that generate test data and help the programmers to write error free codes early in the software development life cycle. In addition, software-testing task requires optimization to save the time and cost of testing. Most of the software engineering problems requires balance between different objectives to solve the problem. Similarly, Testing requires balancing various conflicting test criterion.

Chaos based optimization algorithms are popular for global optimization problem (Yang, Li, & Cheng, 2007) [1]. Various studies on chaos enhanced optimization algorithms suggest its performance competitive when compared to randomized heuristic algorithms (Gandomi *et al.*, 2013) [2]. Chaos based optimization algorithms shows good performance for software testing problems in the recent research study (Pandey & Banerjee, 2017) [17]. This paper presents a case of the applications of chaos-based optimization algorithms for test data optimization. The methodology followed in this study is to formulate the test-data generation problem as a search problem. Followed by parameter tuning of firefly algorithm and genetic algorithm to incorporate chaos in place of randomization. Results are shown for application of chaos tuned algorithm for triangle classification problem and largest of three numbers in software testing.

Related Work

Probably the first application of optimization methods to attack software engineering problem is the work of Miller and Spooner (1976). One of the main important events of 1970s was application of optimization methods for software testing problem (de Freitas & de Souza, 2011) [24]. However, research mainly focus on test data generation until the SEMINAL workshop was organized to discuss the issues such as optimization methods in software engineering (Harman & Jones, 2001b) [25].

Harman & Jones (2001a) [22] coined the word search-based software engineering (SBSE) that reformulates software engineering problems as a search problem. Since the inception of SBSE, There is tremendous and exponential growth of publications encompassing SBSE research over years (Harman, Mansouri, & Zhang, 2012) [23]. Afzal, Torkar and Feldt (2009) [26] presents a systematic review of search based approaches for non-function system properties however they consider publications ranging from 1996-2007. In this work, authors carry out literature review of search-based approaches for non-functional system properties from 2007 to 2017. Besides the work of Afzal *et al.* (2012) various review of search based software testing could be found in the work of (McMinn, 2004; Harman *et al.*, 2012; Harman *et al.*, 2015) [23, 28, 27]. In various studies of search-based software testing requirement of more quality research was sought out in the field of non-functional system properties.

Search based methods are efficient in finding the solutions of tough non-linear problems. Not all efficient meta-heuristics are good in providing the global optimum. Moreover, these algorithms are very good in providing local optimum solutions (Yang, 2012) [19]. Various chaos-enhanced meta-heuristics have shown good optimization performance in many studies (Xiang, 2007) [21]. Moreover, Chaotic tunneling is used with various algorithms such as PSO (Alatas *et al.*, 2009) [20], BAT algorithm (Gandomi & Yang, 2014) [29], Artificial immune optimization (Guo & Wang, 2005) [30] and Imperialist competitive algorithm (Talathari *et al.*, 2012).

Bat algorithm is a recent innovation of Yang. This new metaheuristic method is based on the echolocation properties of bats. After the tremendous success of Firefly-algorithm and harmony-search algorithm a new metaheuristic Bat algorithm also shows good efficiency. Recently a modified version of firefly algorithm is proposed for test path sequence generation and optimization. In this work an objective function is used with traversal guidance matrix is generate test optimizations. A cuckoo search based algorithm is also used for test effort estimation and optimization. Firefly is a recent metaheuristics that employs simulated model of social fireflies. This is based on the flashing behavior of firefly. In a research study a new version of firefly algorithm is proposed and is known a random attraction within firefly algorithm.

Problem formulation and chaos based parameter tuning

This section presents a case of chaos enhanced optimization algorithms for test-suite optimization problem framework. Three optimization algorithms are selected based on their past performances on similar problem area. They are firefly algorithm, Particle swarm optimization algorithm and krill herd algorithm and Genetic algorithm. Major optimization parameters of these algorithms are tabulated in the following table.

Table 1: Parameter tuning of various optimization algorithms

Optimization algorithms	Major optimization parameters	Identified parameter to be tuned with chaotic maps	Chaotic map used
Firefly Algorithm	Attractiveness(β), absorption coefficient(γ), Randomization coefficient(α)	α, β, γ	Logistic Map
Genetic Algorithm	Recombination operator(r) and Mutation operator (μ)	r, μ	Logistic map and Gaussian Map

Test suite optimization using chaos enhanced firefly algorithm

Chaos based Firefly algorithms is the latest variant of the standard firefly algorithm. Firefly algorithm is a meta-heuristic falling into the category of swarm intelligence is the most competitive for global optimization problem according to the recent studies (Pandey & Banerjee, 2017) [17]. Standard firefly algorithm was developed with certain hypothesis such as all fireflies are attracted towards each other, attractiveness is proportional to brightness, and brightness captures the landscape of objective function. As attractiveness of the fireflies is directly proportional with intensity, attractiveness β varies with distance according to the following equation.

$$\beta = \beta 0 \, exp - \gamma r^2 \quad (1)$$

Where β_0 is the attractiveness at the distance r = 0, γ is the

absorption coefficient and r is the Euclidean distance. Dynamics of fireflies is according to the following equation-

$x_i = x_i + \beta_0 e^{-\gamma r^2}(x_j - x_i) + \alpha \epsilon_i$ (2)

The movement of fireflies is governed by the above equation known as firefly movement algorithm. Movement of fireflies (x_i to x_j) is controlled through the parameters such as attractiveness (β), absorption coefficient (γ), randomization parameter (α), random number drawn from Gaussian distribution (ϵ_i).

Tuning attractiveness parameter with logistic chaotic map

The Logistic map is represented by the following equation. The equation appears in nonlinear dynamics of biological population evidencing chaotic behavior.

$Xk+1 = aXk\ (1-Xk)$ (3)

As previously mentioned that attractiveness and absorption coefficient is one of the major parameter in firefly movement algorithm represented by β and γ respectively. These parameters are tuned with chaotic maps (β/γ=Ck). Also randomization parameter is tuned with chaotic maps.

$\beta_{k+1} = a\ \beta_k(1-\beta_k)$ (4)

$\gamma_{k+1} = a\ \gamma_k(1-\gamma_k)$ (5)

$\alpha_{k+1} = a\ \alpha_k(1-\alpha_k)$ (6)

Problem Formulation

In order to formulate software testing problem as a search problem a proper representation of the problem is necessary. Fitness function captures the test adequacy criteria that has to be improved over different iterations. Let the initial test case is a set of n Test data t_1, t_2, t_3....t_n. various encoding techniques exists such as real or binary. In this study binary encoding scheme is used for encoding the initial test data. Calculation of fitness is based on the following branch distance function that has to be minimized in order to increase branch coverage.

$$b(i) = \begin{cases} 0, & if\ branch\ is\ covered \\ k, & if\ branch\ is\ not\ covered \end{cases} \quad (7)$$

Since the maximum value of branch distance is not known, normalized branch distance is used otherwise.

$$normalize(branch\ distance) = 1 - 1.001^{-branch\ distance}$$

Approach level is a measure of untraversed nodes of the control flow graph with respect to the target path.

$$approach\ level(xi) = \frac{\alpha(xi)}{P(t)} \quad (9)$$

Fitness function formulation

Fitness function is based on three parameters such as approach level, controlling parameter (ε) and branch distance.

$$f(b) = \frac{1}{\varepsilon + approach\ level + \sum_{i=1}^{b} wi.normalizeb(i)} \quad (10)$$

Proposed Algorithm

Algorithm 1 Test case generation using chaotic optimization method

1. Input: Test Program, Fitness Function, stop criteria, target path weights (w_i), β/γ, chaotic parameters (C_k)
2. Output: Optimized Test Suite
3. Initialize initial population of test cases T={t_0, t_1,t_2,…….t_n}
4. evaluate test cases using fitness function

$$f(b) = \frac{1}{\varepsilon + approach\ level + \sum_{i=1}^{b} wi.b(i)}$$

5. t=0
6. while t<max gen
7. for i=1 to n
8. for j=1to n
9. if f(t_j)>f(t_i)
10. Move test case t_i towards t_j according to following equation

$$t_i = t_i + \beta_0 e^{-\gamma r^2}(t_j - t_i) + \alpha \epsilon_i$$

where β/γ = C_k ($\beta_{k+1} = a\ \beta_k(1-\beta_k)$, $\gamma_{k+1} = a\ \gamma_k(1-\gamma_k)$,$\alpha_{k+1} = a\ \alpha_k(1-\alpha_k)$)

11. end if
12. end for
13. evaluate new population
14. end for
15. test case ranking based on fitness values
16. t=t+1
17. end while

Test suite Optimization using Chaos enhanced Genetic Algorithm

Genetic algorithms is based on the modeling of search procedure in the evolution of natural selection. Various research and developments in evolutionary algorithms shows that these algorithms are good in certain conditions for parameter optimization (Deb *et al.*, 2000) [32]. These Algorithms are based on the search process that improves the certain quality criteria of individuals. These individuals are initialized randomly and is evolved for better solutions during time. Recombination results in combining the parental information and mutation are used for innovation in the population of individuals (Goldberg & Holland, 1988) [33].

Let f: $R^n \rightarrow R$ defines the objective function to be optimized. Generally, objective function and fitness function needs not to be a same. Let in the minimization problem fitness function be Φ: $I \rightarrow R$. Here assume that f is always a component of Φ. Here i is an individual and x is object variable. In addition, $\mu \geq 1$ is parent population and $\lambda \geq 1$ is offspring population. A population at generation t consists of individuals $P(t) = \{\vec{i_0}, \vec{i_1}, \ldots \vec{i_\mu}\}$, consists of individuals $\vec{i_n} \in I$ and recombination operator is denoted by $U_{\theta r}$: $I^\mu \rightarrow I^\lambda$. Also mutation operator is defined $M_{\theta m}$: $I^\lambda \rightarrow I^\lambda$. These both operators are controlled by the parameters θ_r and

θ_m respectively for recombination and mutation. Selection operator is defined as $S_{\theta s:}\ I^{\mu+\lambda}\ SI^{\lambda} \rightarrow I^{\mu}$.

Algorithm 2 Standard GA

1. initialize the initial population of candidate solution

$P(0):= \{i_0(0),i_1(0),....i_\mu(0)\}$

1. evaluate $P(0):= \{\varphi(i_0(0)),\varphi(i_1(0)),....\varphi(i_\mu(0))\}$
2. while $\tau(P(t))$ =6 *true* do
3. recombine: $P'(t):= U_{\theta r}(P(t))$
4. mutate: $P''(t):= M_{\theta r}(P(t))$
5. *select:* $P(t + 1):= so,(P"(t) \cup Q)$
6. $t:= t + 1$

Chaos based tuning of mutation operator

$$M_{\theta m:}\ I^{\lambda} \rightarrow I^{\lambda} \quad (11)$$

As mentioned in above equation mutation operator maps the offspring population to a new mutated population by means of a random function. Let P_i is the initial offspring population and P_i` is the mutated population of offsprings.

$$P_i' = P_i + \alpha N[0,1] \quad (12)$$

Where α is randomization parameter and $N[0,1]$ is random number drawn from Gaussian distribution. Chaotic maps are used to replace random number generator such as Gaussian distribution with chaotic maps deterministic sequences.

$$P_i' = P_i + \alpha C_i^k \quad (13)$$

$$\alpha_{k+1} = a\ \alpha_k(1- \alpha_k) \quad (14)$$

Where C_i^k is the chaotic map sequence, α is chaotic parameter

Results

The stopping criteria for the experiments is that if at least on test set is produced that traverse the entire target path or if the max generation is reached. The following parameters are selected for results analysis.

No. of evaluation: The number of evaluations for each method employed.

Time taken: Exhausted time for each method employed to generate the test cases.

Branch coverage: Final branch coverage reached by each algorithm.

Benchmark test programs

Triangle classification problem and largest of three numbers are taken as benchmark programs in order to run our algorithms and find the optimized test cases.

Table 2: Experimental settings and results on triangle classification problem for chaos based firefly algorithm

Input range	Pop Size	No. of generations	Mean Evaluations	Branch Coverage (%)
[1,128]	60	300	345.2	90
[1,256]	60	500	600.3	92
[1,512]	100	250	100.3	94.
[1,1024]	200	200	300.1	100
[1,2048]	300	600	200.6	100
[1,4096]	400	800	500.2	100
[1,8192]	500	1000	400.1	100

Table 3: Experimental results on largest of three numbers program for chaos based firefly algorithm

Input range	Pop Size	No. of generations	Mean Evaluations	Branch Coverage (%)
[1,128]	30	100	98.2	98
[1,256]	50	200	200.5	99
[1,512]	100	450	300.2	93
[1,1024]	200	500	450.1	98
[1,2048]	250	600	700.5	99
[1,4096]	300	800	1000.3	100
[1,8192]	500	1000	2000.2	100

It is also desired that performance of chaos based optimization algorithm are compared on the basis of types of chaotic map. Results of application of different chaotic maps are shown below.

Table 4: Results of chaotic behavior of proposed chaos based firefly algorithm with different attractiveness values for triangle classification problem

Standard constant	Input range	Best	Mean	Median	Average time(sec.)	Average Branch Coverage (%)
Logistic map	[1,1024]	2.32E-03	1.98E-04	8.56E-02	91.87	100
Chebyshev map	[1,1024]	5.36E-02	4.78E-02	5.37E-01	123.78	96
Gauss map	[1,1024]	4.38E-01	3.65E-03	4.37E-01	93.87	100
Iterative map	[1,1024]	5.36E-02	4.67E-01	3.87E-03	123.98	93
Sine map	[1,1024]	6.87E-01	7.15E-02	4.67E-01	134.92	90

Table 5: Experimental results for chaos-based genetic algorithm for triangle classification problem

Range	Pop size	No. of generation	Mean evaluation	Branch coverage
[1,128]	50	150	230.4	78
[1,256]	50	200	320.5	89
[1,512]	80	250	245.6	79
[1,1024]	100	200	200.4	90
[1,2048]	300	400	300.5	95
[1,4096]	300	300	240.5	93
[1,8192]	500	500	300.4	98

Threats to validity

Bioinspired optimization algorithms are randomized algorithms. Therefore any change in number of iterations run will change the output results. This makes the major conclusion threat to validity. Though various parameters such as attractiveness of firefly algorithm is tuned with chaotic parameters using various chaotic maps. The branch coverage of proposed algorithm may decrease with slight modifications in parameters of firefly algorithm. This is the major internal validity threat of the proposed algorithm. Statistical validation tests are not performed in this study. This is major internal validity threat.

Future Work

Due to some limitation not all bio-inspired algorithms are examined in this work so examination of other bio-inspired algorithms that are not discussed in this paper is the future task. Also bio-inspired algorithms are rich in certain parameters so these parameters also need to be examined under chaotic maps. Certain different applications of chaos enhanced optimization algorithms must be the future task of this work.

Conclusion

Chaos based optimization algorithms are popular for global optimization problem. Various studies on chaos enhanced optimization algorithms suggest its performance competitive when compared to randomized heuristic algorithm. Chaos based optimization algorithms shows good performance for software testing problems in the recent research study. This paper presents a case of the applications of chaos based optimization algorithms for test data optimization. In this paper, various parameters of bio-inspired algorithm are tuned with the chaotic maps. Experiments are performed on some benchmark test programs such as triangle classification problem and largest of three numbers program. Attractiveness parameter of firefly algorithm is tuned with logistic map. Also Mutation operator of genetic algorithm is tuned with various chaotic maps and experiments are performed for logistic map and Gaussian map. Major results are shown in the tables and it depicts that incorporation of chaos leads to better performance of bio-inspired algorithms in certain cases.

References

1. Yang D, Li G, Cheng G. On the efficiency of chaos optimization algorithms for global optimization. Chaos Solitons Fractals. 2007;34(4):1366-1375.
2. Gandomi AH, Yun GJ, Yang XS, Talatahari S. Chaos-enhanced accelerated particle swarm optimization. Commun Nonlinear Sci Numer Simul. 2013;18(2):327-340.
3. Myers GJ, Sandler C, Badgett T. The art of software testing. Hoboken (NJ): John Wiley & Sons; 2011.
4. Myers GJ, Sandler C, Badgett T. The art of software testing. Hoboken (NJ): John Wiley & Sons; 2015.
5. Korel B. Automated software test data generation. IEEE Trans Softw Eng. 1990;16(8):870-879.
6. Ferguson R, Korel B. The chaining approach for software test data generation. ACM Trans Softw Eng Methodol. 1996;5(1):63-86.
7. Laski JW, Korel B. A data flow oriented program testing strategy. IEEE Trans Softw Eng. 1983;(3):347-354.
8. Korel B. Automated test data generation for programs with procedures. ACM SIGSOFT Softw Eng Notes. 1996;21(3):209-215.
9. King JC. Symbolic execution and program testing. Commun ACM. 1976;19(7):385-394.
10. Cadar C, Sen K. Symbolic execution for software testing: three decades later. Commun ACM. 2013;56(2):82-90.
11. Sen K, Marinov D, Agha G. CUTE: a concolic unit testing engine for C. ACM SIGSOFT Softw Eng Notes. 2005;30(5):263-272.
12. Harman M, Jones BF. Search-based software engineering. Inf Softw Technol. 2001;43(14):833-839.
13. Clarke J, Dolado JJ, Harman M, Hierons R, Jones B, Lumkin M, *et al.* Reformulating software engineering as a search problem. IEE Proc Softw. 2003;150(3):161-175.
14. Liu B, Wang L, Jin YH, Tang F, Huang DX. Improved particle swarm optimization combined with chaos. Chaos Solitons Fractals. 2005;25(5):1261-1271.
15. Afrabandpey H, Ghaffari M, Mirzaei A, Safayani M. A novel bat algorithm based on chaos for optimization tasks. In: Iranian Conference on Intelligent Systems (ICIS); 2014. p. 1-6.
16. Gandomi AH, Alavi AH. Krill herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul. 2012;17(12):4831-4845.
17. Pandey A, Banerjee S. Test suite optimization using chaotic firefly algorithm in software testing. Int J Appl Metaheuristic Comput. 2017;8(4):41-57. doi:10.4018/IJAMC.2017100103
18. Pandey A, Banerjee S. Test suite minimization in regression testing using hybrid approach of ACO and GA. Int J Appl Metaheuristic Comput. 2018;9(3):88-104. doi:10.4018/IJAMC.2018070105
19. Yang XS. Chaos-enhanced firefly algorithm with automatic parameter tuning. Int J Swarm Intell Res. 2012;2(4):125-136.
20. Alatas B, Akin E, Ozer AB. Chaos embedded particle swarm optimization algorithms. Chaos Solitons Fractals. 2009;40(4):1715-1734.
21. Xiang T, Liao X, Wong KW. An improved particle swarm optimization algorithm combined with piecewise linear chaotic map. Appl Math Comput.

2007;190(2):1637-1645.
22. Harman M, Jones BF. Search-based software engineering. Inf Softw Technol. 2001a;43(14):833-839.
23. Harman M, Mansouri SA, Zhang Y. Search-based software engineering: trends, techniques and applications. ACM Comput Surv. 2012;45(1):11.
24. de Freitas FG, de Souza JT. Ten years of search based software engineering: a bibliometric analysis. In: Int Symp Search Based Software Eng. 2011. p. 18-32.
25. Harman M, Jones BF. The SEMINAL workshop: reformulating software engineering as a metaheuristic search problem. ACM SIGSOFT Softw Eng Notes. 2001b;26(6):62-66.
26. Afzal W, Torkar R, Feldt R. A systematic review of search-based testing for non-functional system properties. Inf Softw Technol. 2009;51(6):957-976.
27. McMinn P. Search-based software test data generation: a survey. Softw Test Verif Reliab. 2004;14(2):105-156.
28. Harman M, Jia Y, Zhang Y. Achievements, open problems and challenges for search based software testing. In: IEEE 8th International Conference on Software Testing, Verification and Validation (ICST); 2015 Apr. p. 1-12.
29. Gandomi AH, Yang XS. Chaotic bat algorithm. J Comput Sci. 2014;5(2):224-232.
30. Guo ZL, Wang SA. The comparative study of performance of three types of chaos immune optimization combination algorithms. Acta Simulata Systematica Sinica. 2005;2.
31. Talatahari S, Azar BF, Sheikholeslami R, Gandomi AH. Imperialist competitive algorithm combined with chaos for global optimization. Commun Nonlinear Sci Numer Simul. 2012;17(3):1312-1319.
32. Deb K, Agrawal S, Pratap A, Meyarivan T. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Int Conf Parallel Problem Solving from Nature. Springer; 2000. p. 849-858.
33. Goldberg DE, Holland JH. Genetic algorithms and machine learning. Mach Learn. 1988;3(2):95-99.