International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 13 (2021) pp. 062-071

Received: 5 Jan 2021; Accepted: 12 May 2021; Published: 13 July 2021

Prioritizing and Minimizing the Test Cases using the Dragonfly Algorithms

Anu Bajaj[1], Ajith Abraham[2]

[1] Machine Intelligence Research Labs,
Auburn, Washington, USA
[1]*er.anubajaj@email.com*
[2] *ajith.abraham@ieee.org*

***Abstract*: Regression testing is a necessary but costly process. It involves re-running all of the test cases each time the software is updated. The resources and time needed for retesting can be decreased by minimizing redundancy and prioritizing the test cases. Furthermore, optimization procedures enhance the efficacy of test case prioritization and minimization. In this research, we have proposed a discrete and combinatorial dragonfly algorithm. In addition, its hybrid version is created with a particle swarm optimization algorithm. The suggested approaches are compared to the random search, genetic algorithm, particle swarm optimization and the bat algorithm. The assessment is done on four subject programs of differing sizes. The simulation results show that the proposed methods are more efficient and effective than the compared algorithms. Furthermore, the hybrid algorithm has a compact distribution as seen by boxplots and interval plots of the average percentage of fault detection and the test minimization percentage.**

***Keywords*:** search-based software testing, combinatorial optimization, dragonfly algorithm, particle swarm optimization, test case prioritization, discrete optimization, nature-inspired algorithms, test case minimization, regression testing

I. Introduction

To preserve the software's quality, the rapid development of software necessitates preliminary testing of updates. Therefore, the software is put through regression testing. It ensures that the modifications aren't prone to errors and the upgrades are compatible with the previous one [1]. However, exhaustive retesting is impractical when the size of the test suite grows exponentially [2]. According to [9], on average, a Google developer tests the software twenty times a day, which is a cumbersome task for extensive software. As a result, it is separated into three ways to address the time and cost constraints: test case minimization (TCM), test case prioritization (TCP), and test case selection (TCS) [1].

TCP offers certain benefits over TCS and TCM. Because it simply ranks the test cases according to their importance for achieving predetermined goals [3]. In contrast, TCS and TCM remove the test cases to reduce the effort and time of retesting. TCS picks only those test cases affected by updated area, and TCM clears the redundant test cases. In other words, they may eliminate several crucial test cases [1].

Sorting a large number of test cases, on the other hand, is time-consuming, making it an NP-hard [3]. Optimization methods are effective in resolving these sorts of issues. Nature-inspired approaches are such methods that imitate natural occurrences to address complicated real-world problems [4]. They are becoming more popular as a result of their evident and easy implementation [2]. According to the source of occurrence, these are grouped into physics/ chemistry-inspired, social phenomena-inspired and biology-inspired methods [4].

The majority of regression testing is done using biology-inspired techniques [5]. However, a small number of researchers have used other nature-inspired approaches [6-7]. In contrast, recently developed algorithms like the dragonfly algorithm are yet to be investigated [2]. Therefore, it motivates us to apply the dragonfly algorithms for TCP and TCM in this research. The following are the work's key contributions:

- To suggest a discrete dragonfly algorithm (DA) for TCP succeeded by redundancy removal, i.e., TCM.
- To enhance the DA's effectiveness by hybridizing it with particle swarm optimization algorithm (PSO), namely, DAPSO.
- To evaluate the algorithms' performance on various subject programs of different sizes using the performance measures: average percentage of fault detection (APFD) and test minimization percentage (TMP).
- To analyze the proposed approaches w.r.t. the PSO, genetic algorithm (GA), BAT algorithm and random search (RS).

Initially, the DA is employed to solve a continuous problem [8]. We discretized DA using a fix-up method to patch up the infeasible values to the feasible ones [9]. It's also been hybridized with PSO, and this upgraded version, DAPSO, outperformed the previous one. Furthermore, the TCM approach is used to eliminate unnecessary test cases [6]. The findings reveal that the suggested methods are successful and efficient in tackling the TCP and TCM problems.

Section 2 discusses the current state of regression testing that utilized nature-inspired methods. The DA is described in Section 3, and the proposed approaches are explained in Section 4. Sections 5 and 6 detail the experimental setup and findings, respectively. Section 7 contains the summary of the research and the future work.

II. Literature Review

The nature-inspired approaches applied in regression testing and the applications of dragonfly algorithms are covered in this section.

Li et al. [10] investigated the search-based approaches' performance and the classic methods for TCP. It was observed that the search-based techniques are less effective than the greedy approach. NSGA-II has been used to reduce the test case size and prioritize the test cases based on event coverage [11]. The influence of operators and parameter settings on the TCP was investigated using GA [12]. It was discovered that the method is affected differently by various selection operators and parameter values. As a result, the fitness function design and parameter selections have a considerable impact on algorithm performance [3].

ACO algorithm was used to reduce the test suite. The experiment performed on Java programs revealed that the ACO performed better than the classic heuristics [13]. Marchetto et al. [14] applied a multi-objective evolutionary algorithm considering the requirement coverage, code coverage and execution cost as the objective functions. The experiments indicated that the proposed method was adequate but not efficient than the baseline approaches for reducing the test suite size.

Recently, researchers have begun to apply relatively new nature-inspired methods, e.g., using the BAT method for TCP yielded encouraging results [15]. Khatibsyarbini et al. [16] suggested a flower pollination algorithm for TCP. The similarity distance model was the fitness function, and the experiment showed that the obtained results were not the same as predicted. Sugave et al. [23] used DA for test suite minimization, and the results were better than systolic GA, greedy and bat algorithms. It prompted us to try out DA for solving TCP.

Mirjalili developed the DA algorithm [8]. It is applied to solve various optimization problems, like feature selection [17], numerical optimization problems [18] and wind power forecasting [19]. Sayed et al. [18] developed a chaotic DA by employing ten chaotic maps for the main parameters of the DA to enhance the convergence speed and efficiency of the algorithm. The experimental results showed that the chaotic gauss map outperformed the other maps. DA was iteratively hybridized with PSO to enhance the exploitation capability of DA. The results were better than the DA, PSO and bat algorithms. Li et al. [19] used the improved DA for wind power forecasting. The algorithm was enhanced with the adaptive learning factor and differential evolution strategy. It was applied to support vector machine for optimizing their parameters for better prediction accuracy.

The positive results of DA applications in various disciplines encouraged us to investigate the algorithm's potential for regression testing. However, to the best of our knowledge, we are the first to suggest DA for TCP. For mapping the method to the discrete combinatorial problem, we employed a fix-up technique [9]. In addition, we have also applied the hybrid of the DA and PSO [17] to improve its performance. The results validated the proposed algorithms' effectiveness. The subsequent section covers the fundamentals of the DA.

III. Dragonfly Algorithm

The dragonfly algorithm is a swarm intelligence based optimization developed by Mirjalili [8]. It is motivated by the static and dynamic swarming of dragonflies. It models the social interaction behavior of dragonflies in navigating, seeking meals, and avoiding opponents to design two crucial stages of optimization: exploration and exploitation [16]. During the exploitation phase, swarms of dragonflies travel in one direction covering long ranges, distracting adversaries [17]. However, dragonflies form tiny groups during the exploration phase and fly back and forth over a short region, searching for food and attracting passing preys [18]. Its working is as follows:

A. Initialization

The initial population is produced at random.

B. Fitness evaluation

The fitness values of the population are evaluated in this step. The most widely used fitness function, APFD, is employed in the method (see section 5.2).

C. Update food and enemy sources

The attraction towards food source and distraction from the enemy source are calculated as:

$$F_i = x^+ - x \quad (1)$$

$$E_i = x^- - x \quad (2)$$

here x is the current position, x^+ and x^- are food and enemy sources.

D. Update the swarming weights, separation, alignment and cohesion

The algorithm employs swarming weights, i.e., separation (s), alignment (a), cohesion (c), food factor (f) and enemy factor (e) to guide the artificial dragonflies for various paths. The separation, alignment, cohesion are given by:

$$S_i = \sum_{i=1}^{n} x - x_i \quad (3)$$

$$A_i = \frac{\sum_{i=1}^{n} v_i}{n} \quad (4)$$

$$C_i = \frac{\sum_{i=1}^{n} x_i}{n} - x \quad (5)$$

here x_i and v_i is the position and velocity of i^{th} dragonfly and n is the number of neighboring dragonflies.

E. Population update

The neighborhood distance is calculated using Euclidian distance among all the dragonflies and select n best dragonflies using eq. (6)

$$r_{ij} = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2} \quad (6)$$

If there is at least one dragonfly in the neighborhood then the position is updated using the vectors: step vector (Δx) and position vector (x). The step vector is analogous to velocity of PSO and is calculated as:

$$\Delta x_{t+1} = (sS_i + aA_i + cC_i + fF_i + eE_i) + w\Delta x_t \quad (7)$$

here t is the current iteration. The position of the dragonfly is updated using the following equation:

$$x_{t+1} = x_t + \Delta x_{t+1} \quad (8)$$

Algorithm 1. Dragonfly Algorithm

Begin
Define the MaxPop, MaxIter
Randomly initialize population and step vector
For t = 1 : MaxIter
  Update the fitness
  Update the swarming weights
  For i = 1 : MaxPop
    Update F_i, E_i, S_i, A_i and C_i using (1) - (5)
    Update neighboring radius using (6)
    If there is at least one dragonfly near it
      Update step and position vectors using (7) - (8)
    Else
      Update position using (9)
    End if
    Fix up the solution
  End for
  Store best solution
  Eliminate the duplicate test cases
End for
End

Algorithm 2. DAPSO Algorithm

Begin
Define the MaxPop, MaxIter
Randomly initialize population and step vector
For t = 1 : MaxIter
----------**start of DA**----------
  For i = 1 : MaxPop
    Calculate the fitness $f(x_i)$
    Update DApbest and DAgbest
  End for
  For i = 1 : MaxPop
    Update the swarming weights
    Update F_i, E_i, S_i, A_i and C_i using (1)-(5)
    Update neighboring radius using (6)
    If there is at least one dragonfly in the vicinity
      Update step and position vectors using (7) - (8)
    Else
      Update position using (9)
    End if
    Fix-up the solution
  End for
----------**end of DA and start of PSO**----------
  Initialize population with DApbest
  Set gbest to DAgbest
  For t = 1 : MaxIter
    For i=1:MaxPop
      Calculate fitness
      Update local (pbest) and global best (gbest)
    End for
    For i=1:MaxPop
      Update velocity and position using (10)-(11)
      Fix up the solution
    End for
  End for
----------**end of PSO**----------
  Store best solution
  Eliminate the duplicate test cases
End for
End

Levy flights are introduced to increase the randomness, exploration and exploitation when it does not found any dragonfly in the neighboring radius. It is given as:

$$x_{t+1} = x_t + Levy\,(d) * x_t \quad (9)$$

F. Stop or repeat

Stop if the algorithm exceeds the defined number of iterations. Otherwise, continue step 2.

IV. The Proposed Approaches

This section explains the discrete and combinatorial DA, the hybrid DAPSO for TCP, and the TCM procedure to remove the redundant test cases.

A. Discrete and Combinatorial DA

The DA was initially designed with the goal of continuous optimization in mind. We have presented a discrete and combinatorial DA since our topic is associated with that. All objects are encoded using permutation, i.e., test numbers, as it is a combinatorial problem. The main difference that occurs between the continuous and combinatorial optimization is in the population update. Here we modify the continuous values to permuted numbers, whereas the original approach works with continuous values [9].

In other words, an adaptation strategy inspired by the asexual reproduction mechanism [20-21] is used to update the population's infeasible solutions. The real numbers are then converted to approximate integer values. Finally, individuals who are out of bounds and duplicates are substituted with don't care conditions (*). The previous solution is then replaced for these items in the same sequence as they appeared in the final solution [9]. For instance, when x= [3,2,4,1,5,6] is updated to y= [3.1,5.7,4.2,3.5,1.3,7.5], y is adjusted to [3,5,4,3,1,7], yielding [3,5,4,*,1,*]. The new offspring acquire the remaining individuals (genetic characteristics) with the assistance of the prior solution (parent) to develop a proper solution as [3,5,4,2,1,6].

B. Discrete and Combinatorial DAPSO

The random initialization and levy flights strengthen the exploration capability of the dragonfly algorithm. However, it discards the fitness values below the global solution and does not keep track of the potential solutions which may converge to global optima. This lack of internal memory may stick the algorithm in local optima or converged slowly. Hence, it is improved by adding internal memory to save the potential solutions.

It is done by keeping track of the best fitness values obtained by the dragonflies during the iterations and saved in $DApbest$ and $DAgbest$, respectively. It helps in avoiding local optima. Furthermore, it is followed by iterative hybridization with PSO. PSO is initialized with DA_{pbest} and DA_{gbest} solutions to exploit these potential areas for better solutions. In this way, the hybrid algorithm can balance the exploration with DA and exploitation with the help of PSO [18]. The position and velocity equations of the PSO are given as:

$$v_{t+1}^i = wv_t^i + c_1 r_1\left(DApbest_t^i - x_t^i\right) + c_2 r_2\left(DAgbest_t^g - x_t^i\right) \quad (10)$$

$$x_{t+1}^i = x_t^i + v_{t+1}^i \quad (11)$$

C. Test Case Minimization

We included a redundancy reduction technique after the algorithm to lower the cost and time budget [6]. In other words, it minimizes the test suite size of the current best solution obtained. It selects only those initial test cases that give complete fault coverage. Moreover, it also tells how precisely the proposed algorithms prioritize the test suite. In other words, whether the algorithm is selecting the critical test cases first or it's a random selection. Algorithm 1 and 2 show the pseudo-codes of the discrete DA and DAPSO.

V. Experimental Settings

This section details the subject programs, performance measures and the algorithms' parameters information used to evaluate the proposed work.

A. Subject Programs

The experiments are carried out on the software benchmark repository's subject programs [22]. These programs range in size from small to large. As a result, they help in checking the performance fluctuation with the test suite size scalability. Table 1 lists the specifics of the subject programs. Additionally, the methods are tested using MATLAB R2017, loaded on an HP laptop with 4GB RAM, Intel i5 CPU, and Windows 10.

B. Performance Measures

TCP and TCM's well-known performance measures are utilized for evaluation, i.e., APFD and TMP as described below:

1) Average Percentage of Fault Detection (APFD)

It calculates the weighted sum of the covered faults based on their test suite location [6]:

$$APFD = 1 - \frac{TF_1 + TF_2 + \cdots + TF_n}{mn} + \frac{1}{2n} \quad (12)$$

here m is the faults covered by the test suite (size n). APFD value ranges in $(0, 100)$, and higher is better.

2) Test Minimization Percentage (TMP)

It is the percentage of the redundant test suite (r) to the original test suite (n) [7].

$$TMP = \frac{r}{n} * 100 \quad (13)$$

Furthermore, the proposed approaches are evaluated in comparison to the GA [12], PSO [7], BAT [15] and RS. Finally, because of the algorithms' stochastic character, 30 runs of each algorithm is performed, and the average value is obtained to measure the performance.

C. Parameter Settings

The parameters play an important role in enhancing the speed and effectiveness of the algorithm [12]. Hence, we have selected the parameters by trial and error method followed by Taguchi design of experiments (see Table 2). This method is used to fine-tune the parameters of the algorithms. It uses the signal to noise ratio (SNR) to check the sensitivity of the controlled factors (signal) against the uncontrollable factors (noise) [9]. Since TCP is the maximization problem, so higher the SNR, the better will be the results.

Case Studies	Subject Programs	LOC	Number of Test Cases	Number of faults
CS1	Present	44591	32	139
CS2	Paint	18376	424	148
CS3	Word	4893	772	224
CS4	Spreadsheet	12791	1172	139

Table 1. Software under test

Approaches	Parameters
GA	pcorss=0.8 pmut=0.1 Selecion= Tournament Crossover: ordered Mutation:swap
PSO	c_1=1.5, c_2=2, w_{max}=0.4, w_{min}=0.8
BAT	r_0=0.001, r_{0min} =0, r_{0max} =1, f_{min} =0, f_{max} =1.5, α=0.9, γ =0.99
DA	s=0.2, a=0.25, c=0.6, f=0.8, e=0.8
DAPSO	s=0.2, a=0.25, c=0.6, f=0.8, e=0.8, c1=2, c2=1.5, w_{max}=0.5, w_{min}=0.9
Common Parameters	MaxPop=100, MaxIter=1000, MaxRuns=30

Table 2. Parameter Values

Approaches	CS1	CS2	CS3	CS4	Tukey group ranking
DAPSO	**90.425**	**93.648**	**95.885**	**94.615**	A
DA	88.783	92.110	94.869	93.358	B
GA	87.516	91.171	93.937	92.292	C
PSO	86.018	89.484	92.493	91.094	D
BAT	84.332	87.496	89.459	90.076	E
RS	82.587	86.774	88.270	88.422	F

Table 3. APFD values comparisons

Case Studies	Source	DF	Adj SS	Adj MS	F-Value	P-Value
CS1	Factor	5	1254.5	250.907	154.72	0
	Error	174	282.2	1.622		
	Total	179	1536.7			
CS2	Factor	5	1014.2	202.848	137.21	0
	Error	174	257.2	1.478		
	Total	179	1271.5			
CS3	Factor	5	1388.2	277.649	187.63	0
	Error	174	257.5	1.48		
	Total	179	1645.7			
CS4	Factor	5	759.9	151.98	104.83	0
	Error	174	252.3	1.45		
	Total	179	1012.2			

Table 4. ANOVA Analysis Of The Algorithms For TCP

VI. Results and Analysis

It experimentally tests the proposed algorithms for TCP and TCM using subject programs. Alternatively, it determines the algorithm's performance over a wide variety of test cases. One-way ANOVA test having a level of significance of 0.05 is also performed to statistically assess the algorithms' performance [9] with the following hypothesis:

Null Hypothesis: $\mu_a = \mu_b$

Alternate Hypothesis: $\mu_a \neq \mu_b$

In other words, the p-value < 0.05 rejects the null hypothesis inferring that the algorithms' means are different. Else, there is a statistically insignificant difference among the algorithms. The results are further analyzed for pairwise comparisons of the means of the algorithms. We have used the Tukey HSD post hoc test has been used for the pairwise comparisons. This test separates the algorithms ranking them according to their mean fitness values. If two algorithms do not share the same letter, then they are statistically different. Else they are statistically insignificant. Moreover, boxplots for APFD values display graphical statistics, and interval plots indicate a 95 % confidence limit of TMP.

A. Test Case Prioritization

Table 3 shows the average APFD values of 30 runs for each case study and the Tukey group ranking. The ANOVA analysis of the groups of each subject program is shown in Table 4. It consists of the degree of freedom, means square error, F-value and the p-values, respectively. In addition, the detailed information of the pairwise comparisons of the algorithms is presented in Table 5, containing the difference of means, standard error of differences, t-values and adjusted p-values for all the case studies.

It is observed that all the nature-inspired approaches are superior to the baseline approach, RS. On the other hand, the comparison amongst the nature-inspired algorithms revealed that the proposed DAs outperform the GA, PSO and BAT in all case studies. With a p-value < 0.05, all algorithms are statistically distinct from one another (see Table 4). The boxplots in Figure 1 graphically presents the distribution of the 30 best values obtained from 30 runs of each algorithm. Compared to other algorithms, the compressed boxplot revealed that the hybrid DAPSO delivers the most stable results. It is evident from the figure, as the program's size grows, DAPSO's boxplot becomes denser than other compared algorithms.

B. Test Case Minimization

Table 6 shows the mean TMP values of all the algorithms and the Tukey group ranking. ANOVA analysis and the Tukey HSD simultaneous difference of the means for pairwise comparisons of the algorithms are presented in Tables 7 and 8. According to table 6, all of the nature-inspired algorithms are superior to RS. The ANOVA test further confirms that algorithms have a statistically different mean. Tukey HSD test reveals that PSO and GA are statistically insignificant (see Table 8). On the other hand, Table 6 shows that the numerical mean value of the GA is higher.

The 95% Confidence Interval plots have been shown in Figure 2, which depicts that the proposed DAPSO outperformed all the other algorithms, and DA is the first runner up followed by GA, PSO, BAT and RS. It's also worth noting that the variance across case studies varies. It might be related to the subject program's peculiarities, such as the fact that the spreadsheet has more redundant test cases than the other programs. It confirms that as the size of the test suite increases, the redundancy increases.

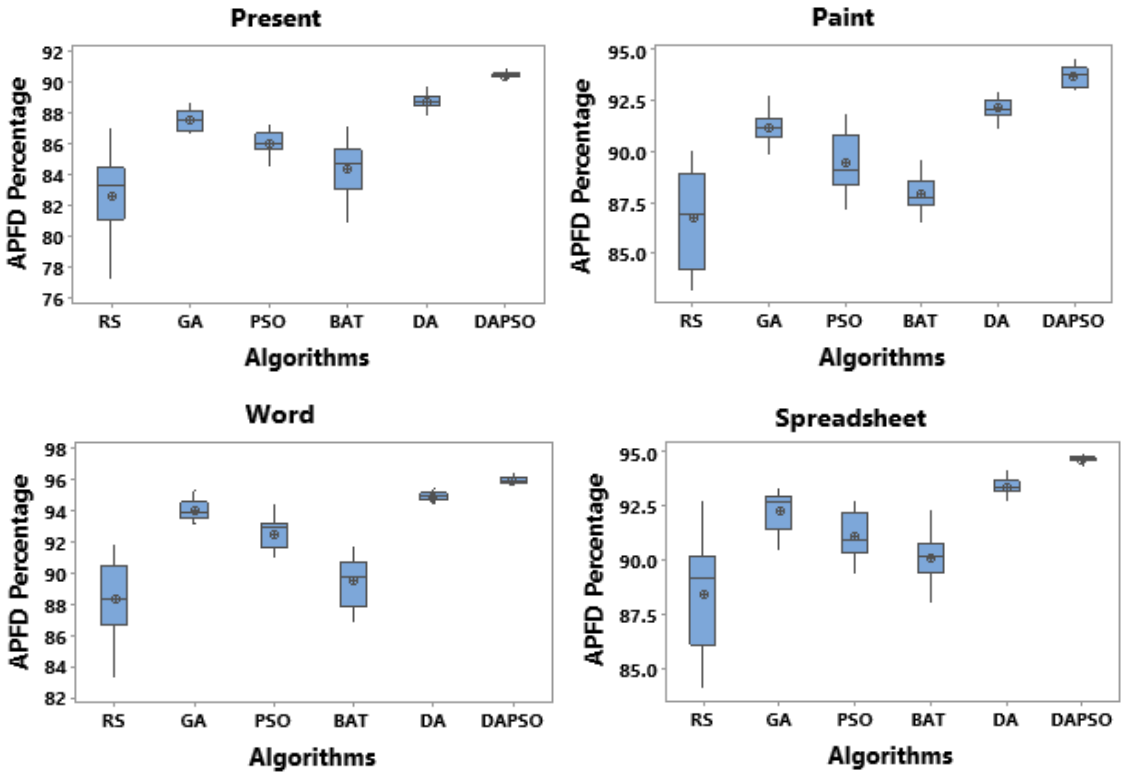

Figure 1. APFD Boxplots of the Case Studies

Case Studies	Difference of levels	Difference of means	SE of Difference	95% CI	T-value	Adjusted p-value
CS1	GA - RS	4.929	0.329	(3.981, 5.878)	14.99	0
	PSO - RS	3.431	0.329	(2.483, 4.380)	10.44	0
	BAT - RS	1.745	0.329	(0.797, 2.694)	5.31	0
	DA - RS	6.196	0.329	(5.247, 7.145)	18.84	0
	DAPSO - RS	7.838	0.329	(6.889, 8.786)	23.84	0
	PSO - GA	-1.498	0.329	(-2.446, -0.549)	-4.55	0
	BAT - GA	-3.184	0.329	(-4.132, -2.235)	-9.68	0
	DA - GA	1.267	0.329	(0.318, 2.215)	3.85	0.002
	DAPSO - GA	2.909	0.329	(1.960, 3.857)	8.85	0
	BAT - PSO	-1.686	0.329	(-2.635, -0.738)	-5.13	0
	DA - PSO	2.765	0.329	(1.816, 3.713)	8.41	0
	DAPSO - PSO	4.406	0.329	(3.458, 5.355)	13.4	0
	DA - BAT	4.451	0.329	(3.502, 5.399)	13.54	0
	DAPSO - BAT	6.092	0.329	(5.144, 7.041)	18.53	0
	DAPSO - DA	1.642	0.329	(0.693, 2.590)	4.99	0
CS2	GA - RS	4.396	0.314	(3.491, 5.302)	14	0
	PSO - RS	2.71	0.314	(1.804, 3.616)	8.63	0
	BAT - RS	1.172	0.314	(0.266, 2.078)	3.73	0.003
	DA - RS	5.335	0.314	(4.430, 6.241)	17	0
	DAPSO - RS	6.874	0.314	(5.969, 7.780)	21.9	0
	PSO - GA	-1.686	0.314	(-2.592, -0.781)	-5.37	0
	BAT - GA	-3.224	0.314	(-4.130, -2.319)	-10.27	0
	DA - GA	0.939	0.314	(0.033, 1.845)	2.99	0.037
	DAPSO - GA	2.478	0.314	(1.572, 3.384)	7.89	0
	BAT - PSO	-1.538	0.314	(-2.444, -0.633)	-4.9	0
	DA - PSO	2.625	0.314	(1.720, 3.531)	8.36	0
	DAPSO - PSO	4.164	0.314	(3.258, 5.070)	13.26	0
	DA - BAT	4.164	0.314	(3.258, 5.069)	13.26	0
	DAPSO - BAT	5.702	0.314	(4.797, 6.608)	18.16	0
	DAPSO - DA	1.539	0.314	(0.633, 2.445)	4.9	0
CS3	GA - RS	5.667	0.314	(4.761, 6.573)	18.04	0
	PSO - RS	4.223	0.314	(3.317, 5.129)	13.45	0
	BAT - RS	1.189	0.314	(0.282, 2.095)	3.78	0.003
	DA - RS	6.598	0.314	(5.692, 7.504)	21.01	0
	DAPSO - RS	7.615	0.314	(6.709, 8.521)	24.25	0
	PSO - GA	-1.444	0.314	(-2.350, -0.538)	-4.6	0
	BAT - GA	-4.478	0.314	(-5.385, -3.572)	-14.26	0
	DA - GA	0.931	0.314	(0.025, 1.837)	2.96	0.04
	DAPSO - GA	1.948	0.314	(1.042, 2.854)	6.2	0
	BAT - PSO	-3.035	0.314	(-3.941, -2.128)	-9.66	0
	DA - PSO	2.375	0.314	(1.469, 3.281)	7.56	0
	DAPSO - PSO	3.392	0.314	(2.486, 4.298)	10.8	0
	DA - BAT	5.41	0.314	(4.503, 6.316)	17.22	0
	DAPSO - BAT	6.427	0.314	(5.520, 7.333)	20.46	0
	DAPSO - DA	1.017	0.314	(0.111, 1.923)	3.24	0.018
CS4	GA - RS	3.87	0.311	(2.973, 4.767)	12.45	0
	PSO - RS	2.672	0.311	(1.775, 3.569)	8.6	0
	BAT - RS	1.654	0.311	(0.757, 2.551)	5.32	0
	DA - RS	4.936	0.311	(4.040, 5.833)	15.88	0
	DAPSO - RS	6.193	0.311	(5.296, 7.090)	19.92	0
	PSO - GA	-1.198	0.311	(-2.095, -0.301)	-3.85	0.002
	BAT - GA	-2.216	0.311	(-3.113, -1.319)	-7.13	0
	DA - GA	1.066	0.311	(0.169, 1.963)	3.43	0.01
	DAPSO - GA	2.323	0.311	(1.426, 3.220)	7.47	0
	BAT - PSO	-1.018	0.311	(-1.915, -0.121)	-3.27	0.016
	DA - PSO	2.264	0.311	(1.367, 3.161)	7.28	0
	DAPSO - PSO	3.521	0.311	(2.624, 4.418)	11.33	0
	DA - BAT	3.282	0.311	(2.385, 4.179)	10.56	0
	DAPSO - BAT	4.539	0.311	(3.642, 5.436)	14.6	0
	DAPSO - DA	1.257	0.311	(0.360, 2.154)	4.04	0.001

Table 5. Tukey post hoc test for difference of means of TCP (Individual confidence level = 99.56%)

Approaches	CS1	CS2	CS3	CS4	Tukey group ranking
DAPSO	53.747	87.325	96.465	97.207	A
DA	50.417	86.674	96.269	96.541	B
GA	46.667	85.967	95.320	95.217	C
PSO	45.375	85.857	95.237	95.054	C
BAT	42.284	85.432	94.598	94.718	D
RS	38.130	84.291	93.622	93.853	E

Table 6. TMP values comparisons

Case Studies	Source	DF	Adj SS	Adj MS	F-Value	P-Value
CS1	Factor	5	4684	936.74	62.79	0
	Error	174	2596	14.92		
	Total	179	7279			
CS2	Factor	5	163.26	32.6519	110.34	0
	Error	174	51.49	0.2959		
	Total	179	214.75			
CS3	Factor	5	167.875	33.575	730.24	0
	Error	174	8	0.046		
	Total	179	175.875			
CS4	Factor	5	227.203	45.4406	948.51	0
	Error	174	8.336	0.0479		
	Total	179	235.539			

Table 7. ANOVA analysis of the algorithms (TCM)

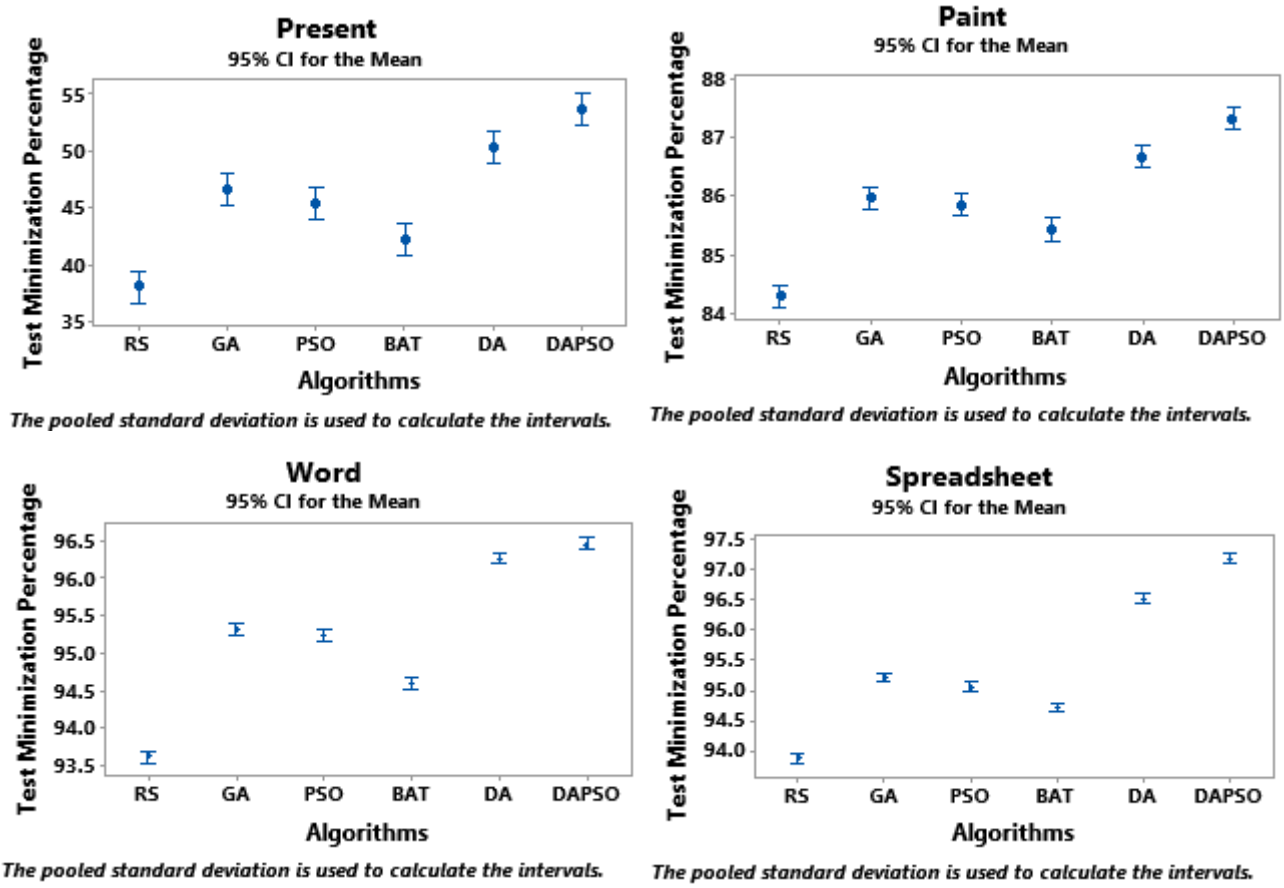

Figure 2. Minimized Suite size interval plots of case studies

Overall, observations from the tables and figures state a massive reduction in the test suite size, i.e., almost 87% to 96% reduction. Though it is required in real-world applications due to extensive test dataset and limited time budget, yet it may lead to the deficiency of some important test cases. Because as the software upgrades, it may discard several test cases that may be necessary to be executed due to the addition of some new requirements related to those test cases.

Case Studies	Difference of levels	Difference of means	SE of Difference	95% CI	T-value	Adjusted p-value
CS1	GA - RS	4.929	0.329	(3.981, 5.878)	14.99	0
	PSO - RS	3.431	0.329	(2.483, 4.380)	10.44	0
	BAT - RS	1.745	0.329	(0.797, 2.694)	5.31	0
	DA - RS	6.196	0.329	(5.247, 7.145)	18.84	0
	DAPSO - RS	7.838	0.329	(6.889, 8.786)	23.84	0
	PSO - GA	-1.498	0.329	(-2.446, -0.549)	-4.55	0
	BAT - GA	-3.184	0.329	(-4.132, -2.235)	-9.68	0
	DA - GA	1.267	0.329	(0.318, 2.215)	3.85	0.002
	DAPSO - GA	2.909	0.329	(1.960, 3.857)	8.85	0
	BAT - PSO	-1.686	0.329	(-2.635, -0.738)	-5.13	0
	DA - PSO	2.765	0.329	(1.816, 3.713)	8.41	0
	DAPSO - PSO	4.406	0.329	(3.458, 5.355)	13.4	0
	DA - BAT	4.451	0.329	(3.502, 5.399)	13.54	0
	DAPSO - BAT	6.092	0.329	(5.144, 7.041)	18.53	0
	DAPSO - DA	1.642	0.329	(0.693, 2.590)	4.99	0
CS2	GA - RS	4.396	0.314	(3.491, 5.302)	14	0
	PSO - RS	2.71	0.314	(1.804, 3.616)	8.63	0
	BAT - RS	1.172	0.314	(0.266, 2.078)	3.73	0.003
	DA - RS	5.335	0.314	(4.430, 6.241)	17	0
	DAPSO - RS	6.874	0.314	(5.969, 7.780)	21.9	0
	PSO - GA	-1.686	0.314	(-2.592, -0.781)	-5.37	0
	BAT - GA	-3.224	0.314	(-4.130, -2.319)	-10.27	0
	DA - GA	0.939	0.314	(0.033, 1.845)	2.99	0.037
	DAPSO - GA	2.478	0.314	(1.572, 3.384)	7.89	0
	BAT - PSO	-1.538	0.314	(-2.444, -0.633)	-4.9	0
	DA - PSO	2.625	0.314	(1.720, 3.531)	8.36	0
	DAPSO - PSO	4.164	0.314	(3.258, 5.070)	13.26	0
	DA - BAT	4.164	0.314	(3.258, 5.069)	13.26	0
	DAPSO - BAT	5.702	0.314	(4.797, 6.608)	18.16	0
	DAPSO - DA	1.539	0.314	(0.633, 2.445)	4.9	0
CS3	GA - RS	5.667	0.314	(4.761, 6.573)	18.04	0
	PSO - RS	4.223	0.314	(3.317, 5.129)	13.45	0
	BAT - RS	1.189	0.314	(0.282, 2.095)	3.78	0.003
	DA - RS	6.598	0.314	(5.692, 7.504)	21.01	0
	DAPSO - RS	7.615	0.314	(6.709, 8.521)	24.25	0
	PSO - GA	-1.444	0.314	(-2.350, -0.538)	-4.6	0
	BAT - GA	-4.478	0.314	(-5.385, -3.572)	-14.26	0
	DA - GA	0.931	0.314	(0.025, 1.837)	2.96	0.04
	DAPSO - GA	1.948	0.314	(1.042, 2.854)	6.2	0
	BAT - PSO	-3.035	0.314	(-3.941, -2.128)	-9.66	0
	DA - PSO	2.375	0.314	(1.469, 3.281)	7.56	0
	DAPSO - PSO	3.392	0.314	(2.486, 4.298)	10.8	0
	DA - BAT	5.41	0.314	(4.503, 6.316)	17.22	0
	DAPSO - BAT	6.427	0.314	(5.520, 7.333)	20.46	0
	DAPSO - DA	1.017	0.314	(0.111, 1.923)	3.24	0.018
CS4	GA - RS	3.87	0.311	(2.973, 4.767)	12.45	0
	PSO - RS	2.672	0.311	(1.775, 3.569)	8.6	0
	BAT - RS	1.654	0.311	(0.757, 2.551)	5.32	0
	DA - RS	4.936	0.311	(4.040, 5.833)	15.88	0
	DAPSO - RS	6.193	0.311	(5.296, 7.090)	19.92	0
	PSO - GA	-1.198	0.311	(-2.095, -0.301)	-3.85	0.002
	BAT - GA	-2.216	0.311	(-3.113, -1.319)	-7.13	0
	DA - GA	1.066	0.311	(0.169, 1.963)	3.43	0.01
	DAPSO - GA	2.323	0.311	(1.426, 3.220)	7.47	0
	BAT - PSO	-1.018	0.311	(-1.915, -0.121)	-3.27	0.016
	DA - PSO	2.264	0.311	(1.367, 3.161)	7.28	0
	DAPSO - PSO	3.521	0.311	(2.624, 4.418)	11.33	0
	DA - BAT	3.282	0.311	(2.385, 4.179)	10.56	0
	DAPSO - BAT	4.539	0.311	(3.642, 5.436)	14.6	0
	DAPSO - DA	1.257	0.311	(0.360, 2.154)	4.04	0.001

Table 8. Tukey post hoc test for difference of means of TCM (Individual confidence level = 99.56%)

As a result, it can be concluded that the DAPSO and DA algorithms outperformed all the compared algorithms for TCP and TCM. Furthermore, the hybrid DAPSO is superior to DA.

VII. Conclusions and Future Work

We have developed a dragonfly algorithm, DA, and its hybrid with PSO, namely, DAPSO for TCP and TCM and proposed algorithms were compared to the random search, GA, BAT, and PSO. The findings showed that the DAPSO approach outperformed existing techniques for APFD and TMP performance metrics. Furthermore, statistical tests proved the proposed algorithm's superiority for test case prioritization and reduction. Boxplots and interval plots also revealed the efficacy of the suggested algorithm DAPSO over its discrete version and other compared algorithms. GA and PSO have statistically insignificant difference for TCM. However, GA performed better in terms of numeric values. In future, the algorithms will be implemented on additional real-world case studies for better validation. Furthermore, applications of the alternative versions of dragonfly algorithms will be explored to enhance the performance further.

References

[1] Yoo, S. and Harman, M., “Regression Testing Minimization, Selection and Prioritization: A Survey”, Software Testing, Verification and Reliability, 22(2), 2012, pp.67-120.

[2] Bajaj, A. and Sangwan, O.P. “A Survey on Regression Testing using Nature-Inspired Approaches”, Proceedings of 4th International Conference on Computing, Communication and Automation (ICCCA), 2018, pp. 1-5, IEEE.

[3] Bajaj, A. and Sangwan, O.P. “A Systematic Literature Review of Test Case Prioritization Using Genetic Algorithms”, IEEE Access, 7, 2019, pp. 126355-126375.

[4] Fister Jr, I., Yang, X.S., Fister, I., Brest, J. and Fister, D., “A brief review of nature-inspired algorithms for optimization.” arXiv preprint arXiv, 2013, pp.1307.4186.

[5] Bajaj, A. and Sangwan, O.P., “Nature-Inspired Approaches to Test Suite Minimization for Regression Testing”, In Computational Intelligence Techniques and Their Applications to Software Engineering Problems CRC Press, 2020, pp. 99-110.

[6] Bajaj, A. and Sangwan, O.P., “Discrete and Combinatorial Gravitational Search Algorithms for Test Case Prioritization and Minimization”, International Journal of Information Technology, 13, 2021, pp. 817-823.

[7] Bajaj, A. and Sangwan, O.P., “Tri-Level Regression Testing using Nature-Inspired Algorithms”, Innovations in Systems and Software Engineering, 17(1), 2021, pp. 1-16.

[8] Mirjalili, S., "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems." Neural Computing and Applications, 27(4), 2016, pp. 1053-1073.

[9] Bajaj, A. and Sangwan, O.P., “Discrete Cuckoo Search Algorithms for Test Case Prioritization”, Applied Soft Computing, 2021.

[10] Li, Z., Harman, M. and Hierons, R.M., “Search algorithms for regression test case prioritization.” IEEE Transactions on software engineering, 33(4), 2007, pp.225-237.

[11] Chaudhary, N. and Sangwan, O.P. "Multi-Objective Test Suite Reduction for GUI Based Software Using NSGA-II." International Journal of Information Technology and Computer Science. 8, 2016, pp. 59-65.

[12] Bajaj, A. and Sangwan, O.P. “Study the Impact of Parameter Settings and Operators Role for Genetic Algorithm Based Test Case Prioritization”, Proceedings of International Conference on Sustainable Computing in Science, Technology and Management, 2019, pp. 1564-1569, Elsevier.

[13] Mohapatra, S. K., and Prasad, S., "Test Case Reduction Using Ant Colony Optimization for Object-Oriented Program.” International Journal of Electrical & Computer Engineering, 5(6), 2015, pp. 2088–8708.

[14] Marchetto, A., Scanniello, G., and Susi, A., “Combining code and requirements coverage with execution cost for test suite reduction.” IEEE Transactions on Software Engineering, 45(4), 2017, pp. 363–390.

[15] Bajaj, A. and Sangwan, O.P., “Test Case Prioritization Using Bat Algorithm”, Recent Advances in Computer Science and Communications, 14(2), 2021, pp. 593-598.

[16] Khatibsyarbini, M., Isa, M.A., Jawawi, D.N., Hamed, H.N.A. and Suffian, M.D.M., Test Case Prioritization Using Firefly Algorithm for Software Testing. IEEE Access, 7, 2019, pp.132360-132373.

[17] KS, S.R. and Murugan, S., "Memory-based hybrid dragonfly algorithm for numerical optimization problems." Expert Systems with Applications, 83, 2017, pp.63-78.

[18] Sayed, G.I., Tharwat, A. and Hassanien, A.E., “Chaotic dragonfly algorithm: an improved metaheuristic algorithm for feature selection.” Applied Intelligence, 49(1), 2019, pp.188-205.

[19] Li, L.L., Zhao, X., Tseng, M.L. and Tan, R.R., “Short-term wind power forecasting based on support vector machine with improved dragonfly algorithm.” Journal of Cleaner Production, 242, 2020, p.118447.

[20] Farasat, A., Menhaj, M.B., Mansouri, T. and Moghadam, M.R.S., “ARO: A new model-free optimization algorithm inspired from asexual reproduction.” Applied Soft Computing, 10(4), 2010, pp.1284-1292.

[21] Mansouri, T., Farasat, A., Menhaj, M.B. and Moghadam, M.R.S., "ARO: A new model-free optimization algorithm for real-time applications inspired by the asexual reproduction." Expert Systems with Applications, 38(5), 2011, pp.4866-4874.

[22] http://www.cs.umd.edu/~atif/Benchmarks/UMD2005b.html

[23] Sugave, S. R., Patil, S. H., and Reddy, B. E. 2017. DDF: Diversity dragonfly algorithm for cost-aware test suite minimization approach for software testing. In 2017 International Conference on Intelligent Computing and Control Systems, 701–707. IEEE.