International Journal of
Software Engineering and Technology

Journal Website: http://ijset.fc.utm.my

ELITISM BASED MIGRATING BIRDS OPTIMIZATION ALGORITHM FOR COMBINATORIAL INTERACTION TESTING

Hasneeza Liza Zakaria[1], Kamal Z. Zamli[2]

[1]School of Computer and Communication Engineering
Universiti Malaysia Perlis(UniMAP)
Kampus Pauh Putra
02600 Arau, Perlis, Malaysia
hasneeza@unimap.edu.my

[2]Faculty of Computer Systems and Software Engineering
Universiti Malaysia Pahang(UMP)
Kampus Gambang, Lebuhraya Tun Razak
26300 Kuantan, Pahang, Malaysia
kamalz@ump.edu.my

***Abstract*— Migrating Birds Optimization Algorithm (MBO) has gained popularity in solving various engineering problems because it yielded a good and consistent result. In this paper, we combined MBO and elitism to solve the Combinatorial Interaction Testing (CIT) problem i.e. to find a set of minimum test case which is an NP-Complete problem. This proposed strategy is the first to utilize population based metaheuristic algorithm i.e. MBO with elitism for solving CIT problem. Elitism is a preservation method that preserves the best population and introduces it back into the next population. Here, we used elitism to preserve the best test cases in order to improve the effectiveness of MBO in generating the minimum set of test cases. This strategy is named as MBO Testing Strategy with elitism (MTS-e). As a comparison with the original MBO we also developed a strategy without elitism, namely MBO Testing Strategy (MTS). MTS yielded a comparable result to the benchmark strategies while MTS-e outperformed most of the benchmarked strategies. The experimental result shows that elitism enhanced the performance of MBO as the mean of the best generated test cases for MTS-e is better than the mean generated by benchmarked strategies.**

Keywords — MBO; elitism; CIT; MTS; MTS-e

1.0 INTRODUCTION

Software plays important role in our life today. Modern humans depend on software to operate many things such as household appliances, gadgetries, transportations, etc. Unfortunately, software has never been perfect and error prone. Software errors could lead to software failures that could cause loss of revenues and even life. Thus, it is important to release software with the most minimum error. In order to minimize software failures, software must be tested before released. There are many stages of software testing and the methods could differ in each stage. Here, we focus on the software test plan stage where we build strategies with Migrating Birds Optimization (MBO) to plan the test with Combinatorial Interaction Testing (CIT) technique.

MBO is a population based nature-inspired meta-heuristic algorithm that mimics the V-formation of migrating birds [1]. The V-formation has been proven by scientist [2-4] to save energy of birds as the energy can be shared among them. The unused neighbor sharing mechanism is unique to MBO and emulates the energy sharing mechanism of the V-formation [5].

MBO Figure 1 starts with the random generation of *n* initial solutions i.e. the number of bird in the V-formation. The best solution is chosen as a leader bird, α and the remaining follower birds, β are alternately distributed to the right and left side of the formation. Leader exchange is done by generating and evaluating the *y* neighbor's solution for the follower birds. Each solution evaluates its *(y-x)* neighbors and *x* unused best neighbors from the bird in front. The best solution will become a new leader and the old leader will move to the end of the formation. When the iteration completed, MBO returns the best solution.

```
Input: n, k, x, m and K
Output: the best solution, α_best
Generate random initial population (n) and put into an imaginary V-shaped structure
iter = 0
while(iter < K)
for(j = 0; j < m; j++)
Improves the leader bird(α) by using best y neighbor
iter = iter + y
for each follower birds solution, β
Improve the β using the best (y-x) neighbor and x unused best neighbor
iter = iter + (y-x)
end for
Forward one of the β solution as leader, α
end while
end for
Return α_best
```

Figure 1 The original MBO algorithm

The original MBO has proven to produce good and consistent results in solving engineering problems [5-8]. The advantages of MBO are that it can enhance the exploration of the search space and offers parallel processing.

The modified MBO has gained popularity since many researchers tried to adapt MBO to solve specific problem domains. These modifications had proven to improve MBO performance [9-14].

Based on the advantages mentioned above, MBO has been chosen for our CIT strategies. Exhaustive testing is impossible due to the combinatorial explosion problem which occurs when the number of configuration and its settings increases, then the number of combinations to be tested also increase. CIT is a technique that tests only the selected number of combinations that are mathematically proven to represent and cover all configurations [15]. Even though MBO has been known to produce good results, but it also has a weakness of early convergence [8]. Hence, elitism is incorporated into MBO to investigate its effectiveness in solving the early convergence problem in MBO.

Elitism has been applied to numerous problems across various filed and had proven to be effective [16-20]. Elitism is a simple preserving mechanism where a number of the best solution from the previous population is carried forward into the next population. Due to its effectiveness, we incorporated elitism into our modified MBO strategy.

In order to investigate the effectiveness of MBO in its original form and with modification, original MBO is implemented to MBO Testing Strategy (MTS) and modified MBO is implemented to MTS with elitism (MTS-e).

This paper is organized as follows. In section 2, the MBO based strategies for CIT are given. In section 3, the parameter tuning of MTS and MTS-e testing strategies with Taguchi method is explained. In section 4, the MTS and MTS-e results are compared with other algorithms. Finally, this paper is concluded in section 5.

2.0 EXISTING WORKS

The number of configurations in software systems nowadays is huge. Thus, the numbers of configuration options expand in huge numbers and therefore, it is impossible to test exhaustively. Thus, there are no sufficient resources and time to test every combination's possible option setting. CIT techniques use sampling method to test selected configurations where each combination's possible option setting for every configuration options can be tested at least once [21].

Existing CIT strategies started with pure computational based approaches like Jenny [22], TConfig [23] and IPOG [24] before the emerging of AI-based approach that mostly used nature inspired metaheuristics algorithm. Nature inspire metaheuristic algorithm have been popular in solving myriad optimization problems in multiple fields such as engineering, networking, data mining and industrial[25].

Meta-heuristics has been popular in solving combinatorial optimization problems because of it's produced a good result. However, according to the No Free Lunch Theorem (NFL) [26; 27] if an algorithm performs well on average for a particular class of problems then it must do worse on average over other classes of problems. This means that even though the meta-heuristics are meant to solve general purpose problems, they cannot perform well on most problems. Hence, there is the need for a problem-specific algorithm that can solve the problem at hand effectively.

In the past 15 years, researchers in CIT have been using nature inspired algorithms in finding the minimum set of test cases. The first 10 years were focused on pairwise and 3-way data generation strategies to test on small size data; mostly with $t \leq 3$ by implementing trajectory based algorithms such as Simulated Annealing(SA), Tabu Search(TS) and Hill Climbing(HC) and classics population based algorithm such as Genetic Algorithm (GA) and Ant Colony Algorithm (ACA)[25]. In the previous 5 years, researchers have been innovatively trying to explore the higher strength ($t > 6$) data generation strategies. This was possible with the creation of new nature inspired algorithms that are mostly population based such as Artificial Bee Colony (ABC), Particle Swarm Optimization Algorithm (PSO), Bee Algorithm (BA) and Bat Algorithm [28]. Population based algorithms have a global exploration and local exploitation mechanism [29]. Hence, they yield a better result compared to trajectory based algorithms.

3.0 THE MTS AND MTS-e STRATEGIES

3.1 Covering Array (CA)

Covering Arrays (CA) are mathematical notations that are applied in t-way testing faults were detected by the interaction of a number of parameters [30]. CA has been used for combinatorial testing for the last 20 years [31]. Uniform strength CA i.e. CA with the same number of configuration values can be represented as CA $(N;t,v^p)$, where N is the final test suite size, t is the interaction strength, v is the uniform configuration value and p is the number of parameters. Kuhn [15] demonstrated that 70% failures could be discovered by 2-way CA and almost all failures could be discovered by 6-way CA. He also concluded that the appropriate t value is between 4 and 6.

CIT methods work by first defining a model of the system's configuration. It is typical for this kind of model to have a set of configuration settings with a small number of options and a set of constraints (if any). The CIT technique produced a set of test suite with the defined model where each combination of system's configuration settings was covered at least once.

Table 1 shows and example of a simple CIT system configuration model i.e. hotel room control system with the same number of configuration values. Table 2 shows the 3-way CA for the system. If we were to test the system exhaustively, there will be 16 number of combinations. CIT technique enabled us to test with only 8 test cases which is a 50% reduction. Therefore, a lot of time and money could be saved when testing a large system because the number of test cases is significantly reduced.

Table 1 Hotel room control system

Auto Lock Door	Curtain	Lighting	Air Conditioner
On	Open	On	High
Off	Close	Off	Low

Table 2 CA$(8;3,2^{4})$ for hotel control system

Auto Lock Door	Curtain	Lighting	Air Conditioner
Off	Open	Off	Low
Off	Close	On	Low
Off	Close	Off	High
On	Open	Off	High
On	Open	On	Low
Off	Open	On	High
On	Close	On	High
On	Close	Off	Low

3.2 MTS and MTS-e

The OTAT strategy has been popular in solving CIT problems [32]. The OTAT strategy aims to generate one test case at a time until the each combination of configuration settings is covered. The algorithm begins by initializing a set of target combinations of a configuration. A test case that covers as many target combinations i.e. has the maximum weight is generated. Then, the covered target combinations will be removed. The loop terminates when all test cases generated covers all target combinations.

The MTS and MTS-e combines the OTAT strategy with MBO. Basically OTAT an interaction elements list (e-list) is constructed first by MTS. The exhaustive tuple combinations of each p-valued accepted input are stored in an interaction elements list (e-list). The current test case (cur_tc) is generated first, then the neighbor test case (nbr_tc) is generated as a local search test case by MBO. Both test cases weight are compared and the one with the largest weight will be selected as the best solution. The pair interactions of tuple combinations corresponding to the best test solution will be eliminated from the e-list. Lastly, the best solution will be inserted into the test suite. The OTAT based MTS/MTS-e strategy is depicted in the flowchart Figure 2.

The MTS-e Figure 3 applied elitism to solve the quick convergence problem of MBO. Elitism is as a simple mean to preserve the best solutions from a population and then introduces them into the next population [33]. An elitist storage was created to keep the good solutions from the previous run. In the first iteration, the elitism algorithm run and save a certain percentage of good solutions into the elitist array. Afterward, in the second iteration, the elitist from the first run will be inserted into the next population. Then, the second elitists will be kept in the elitist array. This cycle continues until the algorithm completed.

Input: *n, k, x, m and K*
Output: *the best solution, a_{best}*

1. *Generate random initial population (n) and put into an imaginary V-shaped structure*
2. *Generate a neighbor flock*
3. *If(!initial flock)*
4. *Add elitist to the flock*
5. *iter = 0*
6. *while(iter < K)*
7. *for(j = 0; j < m; j++)*
8. *Improves the leader bird(a) by using best y neighbor*
9. *iter = iter + y*
10. *for each follower birds solution, β*
11. *Improve the β using the best (y-x) neighbor and x unused best neighbor*
12. *iter = iter + (y-x)*
13. *end for*
14. *Forward one of the β solution as leader, a*
15. *end while*
16. *end for*
17. *Send the remaining one third of the best solutions to the elitist storage*
18. *Return a_{best}*

Figure 3 The MTS-e algorithm

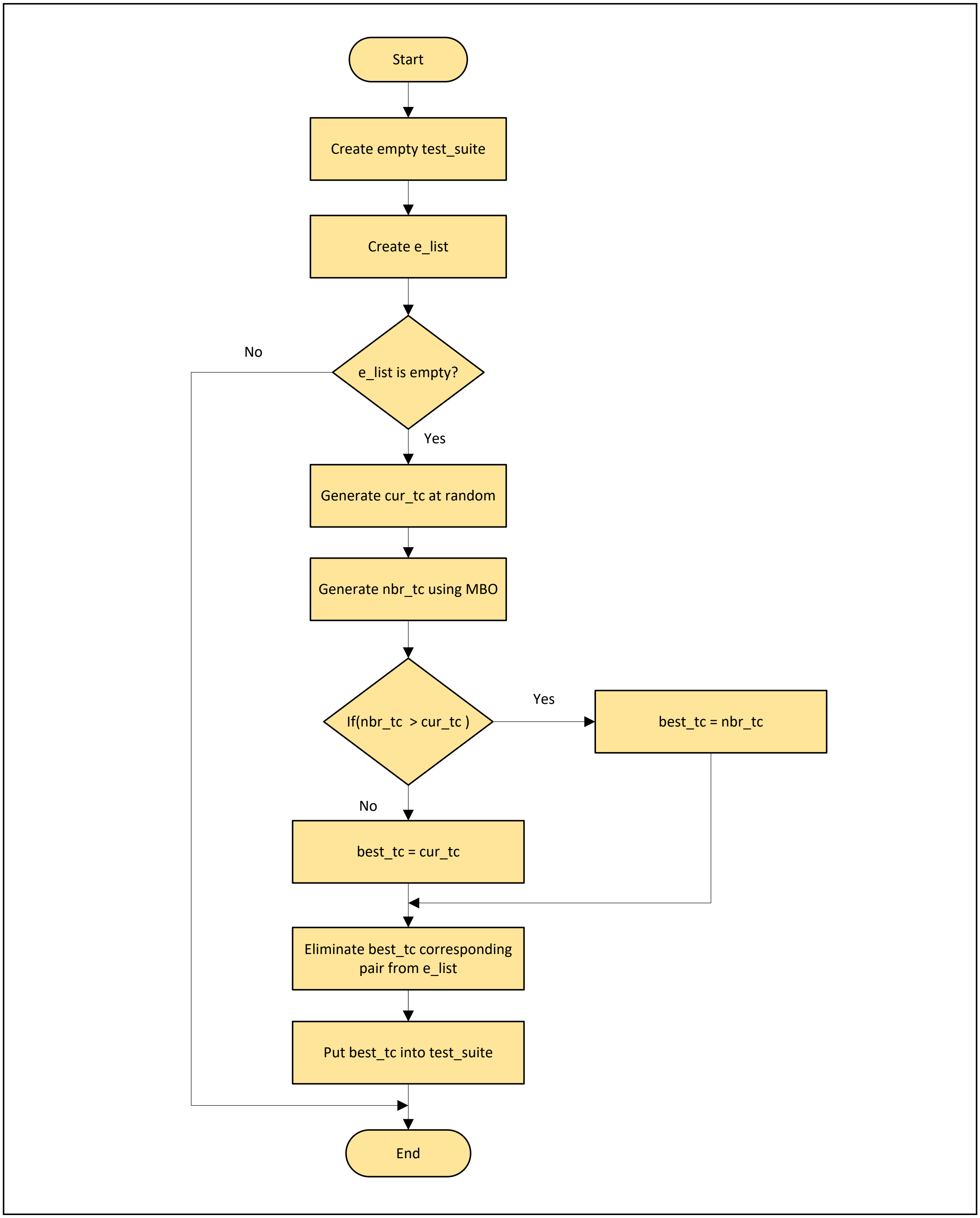

Figure 2 The OTAT based MTS/MTS-e flowchart

Elitism stores a number of the best test cases from a run. Then, these test cases were put back into the next run. Thus, introduce diversity of the best solutions into the population. Two random test cases were chosen from the population and the one with more weight is chosen as the best test case. This best test case will replace the poorest test case in the previous population. Then, the best test cases from the previous population were added to the elitist array. This increased the number of the best solutions in the population and increases the probability of getting better test cases. The addition of elitism mechanism into the original MBO is shown in Figure 4.

```
1.  Specify the elitism percentage; e_percentage
2.  Declare and initialize the elitist array; elitist_array
3.  int i = 0;
4.  while(i < e_percentage)
5.  Get  2 random test case from population; a,b  and
    check their weight
6.  if(weight_a > weight_b)
                    best test case = a;
                          else
                    best test case = b;
7.  Get the poorest test case from the previous
    population; m
8.  Check weight of m
9.  if(best test case weight > weight_m)
                    best test case = m;
10. Add best test case to elitist_array
11. i++;
12. end while
```

Figure 4 The elitism algorithm for MTS-e

3.3 Parameter Tuning of the MTS and MTS-e

Taguchi method is often used in engineering field for parameter tuning [34-36]. It is also being applied to tune parameters for meta-heuristic algorithms such as Particle Swarm Optimization Algorithm (PSO) [37], Simulated Annealing (SA) [38] and Genetic Algorithm (GA) [39]. We chose the Taguchi method to tune the parameters of the t-way testing strategies with MBO as it is proven as a suitable method to tune MBO Algorithm by Niroomand *et al.* [12] in solving the closed looped layout problem in manufacturing systems.

First, we identified the control factors and noise factors of the strategies. The control factors are the parameters that we can control and want to tune. Whereas the noise factors are the parameters that we cannot control and do not need to tune. Here, the control factors are n, k and m where n is the number of initial solutions, k is the number of neighbor solutions and m is the number of tours [12]. The other factors, i.e. x and K where x is the number of shared neighbor solutions and K is the number of iterations are treated as the noise factor by the following relations, $x = k$ and $K = knm$. The x value is set to 1 as suggested by Duman [1]. Therefore the value of k should be set to $2x + 1$ which is 3. Thus, the 2 control factors for MTS are n and m and 3 control factors for MTS-e are m, n and elitism, e.

The Degrees of Freedom (DOF) are computed for MTS and MTS-e before selecting a suitable orthogonal array. The total DOF for MTS is 7, which is (2x3)+1, where 1 is the DOF Mean Value. As for MTS-e, the total DOF is 9, which is (2x4)+1, where 1 is the DOF Mean Value. Thus, the most suitable orthogonal array for experimentation is L9 array as shown in Table 3.

Table 3 L9 Orthogonal Array for MTS/MTS-e

Experiment No.	**Control Factors for MTS/MTS-e**		
	n	***m***	***e***
1	25	1	0.33
2	25	3	0.50
3	25	10	0.67
4	51	1	0.67
5	51	3	0.33
6	51	10	0.50
7	101	1	0.50
8	101	3	0.67
9	101	10	0.33

Three representatives covering arrays are selected for parameter tuning i.e. $CA(N;2,3^3 4^2 5^2)$; $CA(N;2,5^{10})$ $CA(N;3,4^7)$. Nine experiments were run for each covering arrays and each experiment was run 20 times.

All test cases from the experiments are normalized in the range of [0, 1] since the size of test cases varies for different CA. Feature scaling was used for data normalization as in [40].

$$X_{i,0\ to\ 1} = \frac{X_i - X_{Min}}{X_{Max} - X_{Min}} \tag{1}$$

Where,
X_i = each test case size generated
X_{Min} = the minimum test case size
X_{Max} = the maximum test case size
$X_{i,\ 0\ to\ 1}$ = normalized test case between 0 and 1

The mean of $CA(N;2,5^{10})$, $CA(N;3,4^7)$ and $MCA(N;2,3^3,4^2,5^3)$ is then compared and analyzed to produce the main effects plot in Fig. 3 and Fig. 4. The loss function is a statistical method that calculates the losses incurred when the performance measured did not meet the target value [41]. The value of the loss function is measured in the form of signal-to-noise (SN) ratio. There are three types of SN ratio i.e. smaller-the-better, nominal-the-best and larger-the-better. In our case the smaller-the-better is chosen because the ideal target value should be as small as possible.

A larger SN ratio indicates a better performance. Referring to Figure 5 and Figure 6, the highest mean of the SN ratio for both MTS and MTS-e shows the best parameter settings to obtain the most minimum test case size.

However, in order to reduce the long computational time, the e value of MTS-e is set to 0.33 instead of 0.67. Thus, the recommended parameter setting is as in Table 4.

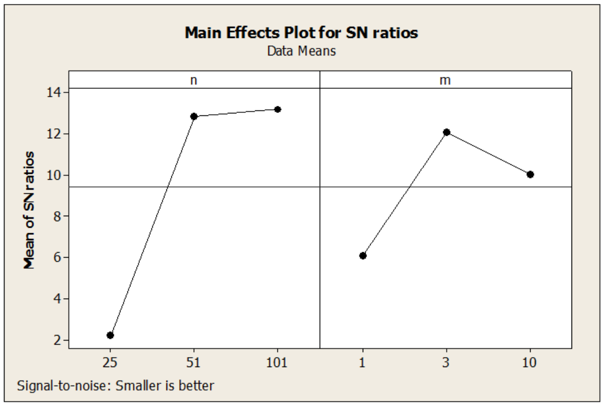

Figure 5 SN plot for MTS

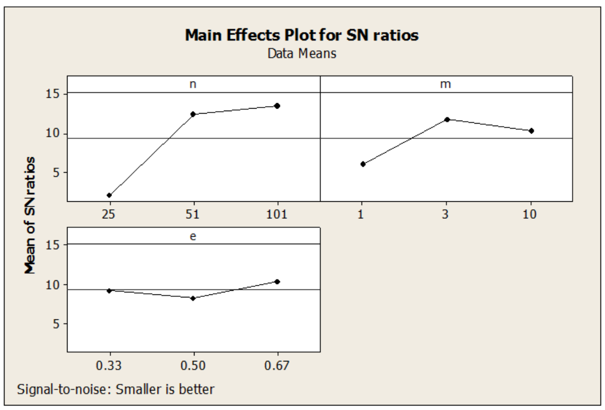

Figure 6 SN plot for MTS-e

Table 4 The recommended parameter settings

	n	*m*	*k*	*e*
MTS	101	3	3	-
MTS-e	101	3	3	0.33

4.0 RESULT AND DISCUSSION

Several experiments were conducted to find the effectiveness of incorporating elitism into MBO by benchmarking MTS-e with several popular strategies alongside MTS. Those strategies are PSO, IPOG, TVG, PICT, TConfig and Jenny.

Due to the long execution time, we were experimenting only with strength 3 and 4 only. In order to measure effectiveness, each experiment is performed 20 times. Then, the best test case (minimum test case) and its mean are reported for some of the experiments. The experiments were run on a Pentium i7 Core processor 3.40 GHz, 4.00 GB Ram and on Windows 8.

Table 5 presents the result for CA(N;3,3^p) for different strategies where p varies from 4 to 10 and also for CA(N;4, 3^p) where p varies from 5 to 10. The best test case size of all strategies was compared. The result marked in bold signified the best result for a particular strategy.

The first thing that can be observed in Table 5 is that the metaheuristics strategies i.e. PSO, MTS and MTS-e managed to outperform the computational based strategies. MTS-e produce smaller size test case than the other benchmarked strategies in almost all configurations. Some of their means also smaller compared to the other strategies. Statistical analyses were also conducted to see the paired significant difference of benchmarked strategies against MTS and MTS-e.

The effectiveness of a method can be verified using statistical analysis of two and multiple method's comparisons over multiple data sets [42]. Statistical analyses were conducted for all the obtained results in Table 5 based on multiple pairwise comparisons with 95% confidence level (i.e. $\alpha = 0.05$) to find the significant difference of the strategies. The non-parametric Friedman test and Wilcoxon Rank-Sum test were used. The non-parametric methods were chosen because the results are not normally distributed and the sample size was small [43]. IBM SPSS Statistics 22 was used to run the statistical analysis.

Friedman test was conducted to find the mean rank of the strategies. The test rendered a chi-square ($\chi 2$) value of 80.121 with $p < 0.001$. Table 10 shows the mean rank of each strategy and MTS-e proven to have the smallest mean rank. Thus, MTS-e outperformed the other strategies with the minimum mean rank of 1.23. MTS came in second with a mean rank of 2.00.

Table 10 The Friedman's Mean Rank

Strategies	Mean Rank
Jenny	5.08
TConfig	5.42
PICT	5.46
TVG	6.15
IPOG	7.88
PSO	2.77
MTS	2.00
MTS-e	1.23

The Wilcoxon Rank-Sum test was run to see the detail comparison of each paired strategies. The null hypothesis (H_0) suggested the best test case size of MTS and MTS-e with the best test case size of the benchmarked strategies do not have any significant differences. The alternative hypothesis (H_1) is that there is a significant difference between their mean. The H_0 is rejected when the sum of the negative ranks (P^-) is less than or equal to the critical value of Wilcoxon signed-rank test (P_α) i.e. $P^- \leq P_\alpha$.

Table 5 Result for $CA(N;3,3^p)$ and $CA(N;4,3^p)$ with varying p values

t	p	Jenny	TConfig	PICT	TVG	IPOG	PSO	Mean	MTS	Mean	MTS-e	Mean
3	4	34	32	34	34	39	**27**	**29.3**	29	31.20	28	31.65
	5	40	40	43	41	43	39	41.37	**38**	**40.35**	**38**	40.95
	6	51	48	48	49	53	45	46.76	43	45.90	**42**	**45.00**
	7	51	55	51	55	57	50	52.2	49	50.75	**48**	**50.80**
	8	58	58	59	60	63	54	56.76	**52**	54.60	**52**	**54.35**
	9	62	64	63	64	65	58	60.3	57	58.55	**56**	**57.85**
	10	65	68	65	68	68	62	63.95	60	61.50	**59**	**61.20**
4	5	109	97	100	105	115	96	97.83	96	100.15	**94**	99.85
	6	140	141	142	139	181	133	135.31	**132**	**135.35**	**132**	135.70
	7	169	166	168	167	185	155	158.12	155	157.20	**154**	**157.00**
	8	187	190	189	192	203	175	176.94	174	175.90	**172**	**175.57**
	9	206	213	211	215	238	195	198.72	191	194.30	**190**	**193.14**
	10	221	235	231	233	241	210	212.71	208	210.00	**208**	**208.75**

Table 6 Result of Wilcoxon signed-rank test for MTS when t = 3

MTS vs. Jenny	MTS vs. PICT	MTS vs. Tconfig	MTS vs. IPOG	MTS vs. PSO	MTS vs. MTS-e
Reject H_0 with $P^- = 0, P_\alpha = 3$ $P^- \leq P_\alpha$	Reject H_0 with $P^- = 0, P_\alpha = 3$ $P^- \leq T_\alpha$	Reject H_0 with $P^- = 0, P_\alpha = 3$ $P^- \leq P_\alpha$	Reject H_0 with $P^- = 0, P_\alpha = 3$ $P^- \leq P_\alpha$	Cannot Reject H_0 with $P^- = 4, P_\alpha = 3$ $P^- > P_\alpha$	Cannot Reject H_0 with $P^- = 15,$ $P_\alpha = 0$ $P^- > P_\alpha$

Table 7 Result of Wilcoxon signed-rank test for MTS-e when t = 3

MTS-e vs. Jenny	MTS-e vs. PICT	MTS-e vs. Tconfig	MTS-e vs. IPOG	MTS-e vs. PSO	MTS-e vs. MTS
Reject H_0 with $P^- = 0, P_\alpha = 3$ $P^- \leq T_\alpha$	Reject H_0 with $P^- = 0, P_\alpha = 3$ $P^- \leq P_\alpha$	Reject H_0 with $P^- = 0, P_\alpha = 3$ $P^- \leq P_\alpha$	Reject H_0 with $P^- = 0, P_\alpha = 3$ $P^- \leq P_\alpha$	Reject H_0 with $P^- = 1, P_\alpha = 3$ $P^- \leq P_\alpha$	Cannot Reject H_0 with $P^- = 15, P_\alpha = 0$ $P^- > P_\alpha$

Table 8 Result of Wilcoxon signed-rank test for MTS when t = 4

MTS vs. Jenny	MTS vs. PICT	MTS vs. Tconfig	MTS vs. IPOG	MTS vs. PSO	MTS vs. MTS-e
Reject H_0 with $P^- = 0, P_\alpha = 2$ $P^- \leq P_\alpha$	Reject H_0 with $P^- = 0, P_\alpha = 2$ $P^- \leq P_\alpha$	Reject H_0 with $P^- = 0, P_\alpha = 2$ $P^- \leq P_\alpha$	Reject H_0 with $P^- = 0, P_\alpha = 2$ $P^- \leq P_\alpha$	Not enough data for MTS	Not enough data for MTS

Table 9 Result of Wilcoxon signed-rank test for MTS-e when t = 4

MTS-e vs. Jenny	MTS-e vs. PICT	MTS-e vs. Tconfig	MTS-e vs. IPOG	MTS-e vs. PSO	MTS-e vs. MTS-e
Reject H_0 with $P^- = 0, P_\alpha = 2$ $P^- \leq P_\alpha$	Reject H_0 with $P^- = 0, P_\alpha = 2$ $P^- \leq P_\alpha$	Reject H_0 with $P^- = 0, P_\alpha = 2$ $P^- \leq P_\alpha$	Reject H_0 with $P^- = 0, P_\alpha = 2$ $P^- \leq P_\alpha$	Reject H_0 with $P^- = 0, P_\alpha = 2$ $P^- \leq P_\alpha$	Not enough data for MTS

The results of Wilcoxon signed-rank test in Tables 6 through Tables 8 show that the MTS and MTS-e outperformed the computational based strategies in all cases i.e. Jenny, PICT, TConfig, TVG and IPOG. On the other hand, the result for the PSO is better than MTS when t=3, contrary to MTS-e because it is better than PSO when t=3. It is also the same when t=4, where MTS-e is better than PSO. MTS and MTS-e were also compared for t=3 and MTS-e outperformed MTS. Unfortunately, the sample size is too small to compare PSO with MTS when t=4. The MTS and MTS-e also cannot be compared when t=4 because of the same reason.

This MTS-e's result shows that elitism did a great job in enhancing the number of the best solutions to be chosen in the population and produced smaller test cases compared to the benchmarked strategies. The mean of test cases after 20 runs also better than the benchmarked strategies.

5.0 CONCLUSION

In this paper, we analyzed the effectiveness of incorporating elitism into MBO in CIT strategy, denoted as MTS-e. It is achieved by preserving 33 percent of the best test cases from the previous population and then introduced them back again into the next population. We compared MTS-e with the original MBO for CIT, denoted MTS and also with a few other benchmark strategies i.e. Jenny, TConfig, PICT, TVG, IPOG and PSO. The result shows MTS-e produced a better result compared to the other strategies. This proves that elitism helps in enhancing the capability of MBO by improving the next population with the best solutions from the previous population. However, the limitation of these strategies is that they are slower than the computational based strategies. This problem can be rectified if they were run on a faster computer. Researches in nature inspired metaheuristic algorithms are very promising and as part of future work, the MTS and MTS-e could be enhanced to support constraint CIT or sequence based strategy.

Acknowledgement

This work is funded by the ERGS Grant entitled: A Computational Strategy for Sequence based t-way Testing.

References

[1] Duman, E., Uysal, M., and Alkaya, A. F. 2012. Migrating birds optimization: A new metaheuristic approach and its performance on quadratic assignment problem. *Information Sciences* 217: 65-77.

[2] Voelkl, B., Unsöld, M., Usherwood, J. R., and Wilson, A. M. 2015. Matching times of leading and following suggest cooperation through direct reciprocity during v-formation flight in ibis. Proceedings of National Academy of Sciences of the United States of America. 17 February 2015. 2115 – 2120.

[3] Cutts, C. J., and Speakman, J. R. 1994. Energy savings in formation flight of pink-footed geese. *Journal of Experimental Biology* 189: 251-61.

[4] Bajec, I. L., and Heppner, F. H. 2009. Organized flight in birds. *Animal Behaviour* 78(4): 777-89.

[5] Makas, H., and Yumusak, N. 2016. System identification by using migrating birds optimization algorithm: A comparative performance analysis. *Turkish Journal of Electrical Engineering & Computer Sciences* 24:1879 – 900.

[6] Tongur, V., and Ülker, E. 2014. Migrating birds optimization for flow shop sequencing problem. *Journal of Computer and Communications* 2(4):142-47.

[7] Ramanathan, L., and Ulaganathan, K. 2014. Nature-inspired metaheuristic optimization technique-migrating bird ' s optimization in industrial scheduling problem. *SSRG International Journal of Industrial Engineering (SSRG-IJIE)* 1(3):1-6.

[8] Duman, E., Buyukkaya, A., and Elikucuk, I. 2013. A novel and successful credit card fraud detection system implemented in a turkish bank. *2013 IEEE 13th International Conference on Data Mining Workshops*. Dallas, Texas. 7-8 December 2013.162-171.

[9] Soto, R., Crawford, B., Almonacid, B., and Paredes, F. 2016. Efficient parallel sorting for migrating birds optimization when solving machine-part cell formation problems. *Scientific Programming, Hindawi Publishing Corporation*:1-39.

[10] Shen, L. W., and Asmuni, H. 2015. Migrating birds optimization in solving university timetabling problem. *Jurnal Teknologi* 72(1): 89–96.

[11] Pan, Q.-K., and Dong, Y. 2014. An improved migrating birds optimisation for a hybrid flowshop scheduling with total flowtime minimisation. *Information Sciences* 277: 643-655.

[12] Niroomand, S., Hadi-Vencheh, A., Şahin, R., and Vizvari, B. 2015. Modified migrating birds optimization algorithm for closed loop layout with exact distances in flexible manufacturing systems. *Expert Systems with Applications* 42: 6586-6597.

[13] Makas, H., and Yumusak, N. 2013. New cooperative and modified variants of the migrating birds optimization algorithm. International Conference on Electronics, Computer and Computation, at Ankara, Turkey. 7-9 November 2013. 176-179.

[14] Lalla-ruiz, E., Exp, C., and Armas, J. D. 2014. Migrating birds optimization for the seaside problems at maritime container terminals. *Journal of Applied Mathematics*:1-21.

[15] D. Richard Kuhn, Wallace, D. R., and Jr., A. M. G. 2004. Software fault interactions and implications for software testing. *IEEE Transactions on Software Engineering* 30 (6): 1-3.

[16] Moreto, R. A. d. L., Gimenez, S. P., and Thomaz, C. E. 2013. Analysis of a new evolutionary system elitism for improving the optimization of a cmos ota. 2013 BRICS Congress on Computational Intelligence & 11th Brazilian Congress on Computational Intelligence. Recife, Brazil. 8 – 11 September 2013. 628-637.

[17] McLaughlin, M., and Wineberg, M. 2014. The effects of elitism on spatial coevolutionary gas. *ALIFE 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems*. New York, NY, USA. 30 July – 2 August 2014. 924-931.

[18] Nayeem, M. A., Rahman, M. K., and Rahman, M. S. 2014. Transit network design by genetic algorithm with elitism. *Transportation Research Part C* 46: 30-45.

[19] Wu, M.-S. 2014. Particle swarm optimization based on elitism for fractal image compression. In *Lecture notes in electrical engineering*, edited by J. J. e. al. Switzerland: Springer International Publishing.

[20] Xiang, Y., Zhou, Y., and Liu, H. 2015. An elitism based multi-objective artificial bee colony algorithm. *European Journal of Operational Research* 245: 168-93.

[21] Yilmaz, C., Fouch, S., x00E, Cohen, M. B., Porter, A., Demiroz, G., and Koc, U. 2014. Moving forward with combinatorial interaction testing. Computer 47 (2):37-45.

[22] B.Jenkins. Jenny test tools. Available from http://www.burtleburtle.net/bob/math/jenny.html.

[23] Williams, A. 2016. Tconfig]. Available from http://www.site.uottawa.ca/~awilliam/.

[24] Lei, Y., Kacker, R., Kuhn, D. R., Okun, V., and Lawrence, J. 2007. Ipog: A general strategy for t-way software testing. Paper read at Proceedings of the International Symposium and Workshop on Engineering of Computer Based Systems.

[25] Zamli, K. Z. 2015. Software testing inspiration from nature. Kuantan, Pahang: Penerbit Universiti Malaysia Pahang.

[26] Wolpert, D. H., and McReady, W. G. 2007. No free lunch teorem for optimization. IEEE Transactions on Evolutionary Computing 1 (1).

[27] Wolpert, D. H. 2012. What the no free luch theorems really mean: How to improve search algorithms. Santa Fe Institute.

[28] Xing, B., and Gao, W.-J. 2014. Intelligent systems reference library 62 innovative computational intelligence : A rough guide to 134 clever algorithms. Switzerland: Springer International Publishing.

[29] Beheshti, Z., Mariyam, S., and Shamsuddin, H. 2013. A review of population-based meta-heuristic algorithm. International Journal of Advances in Soft Computing and its Applications 5 (1):1-35.

[30] Colbourn, C. J. 2011. Covering arrays and hash families. In *Information security, coding theory and related combinatorics*. IOS Press.

[31] Zhang, J., Zhang, Z., and Ma, F. 2014. Automatic generation of combinatorial test data. London: Springer-Verlag.

[32] Zamli, K. Z., Othman, R. R., and Zabil, M. H. M. 2011. On sequence based interaction testing. *IEEE symposium on computers & informatics*. Kuala Lumpur, Malaysia. 20-23 March 2011. 662-667.

[33] Jong, D. 1975. Analysis of the behaviour of a class of genetic adaptive systems. PhD thesis, University of Michigan, Ann Anbor.

[34] Turning, W., and Steel, E. N. 2013. Taguchi method and anova: An approach for process parameters optimization of wet turning operation while turning en 353 steel. *International Journal of Advanced Research in Engineering and Technology (IJARET)*:1-7.

[35] Shirazi, A. N., B, Y., Gholamian, S. A., and S, R. 2013. Application of taguchi experiment design for decrease of cogging torque in permanent magnet motors. *International Journal on Computational Science & Applications* 3: 31-38.

[36] Panda, A. K., and Singh, R. K. 2013. Optimization of process parameters by taguchi method : Catalytic degradation of polypropylene to liquid fuel. *International Journal of Multidisciplinary and Current Research 2(4)*: 50-58.

[37] Wang, H., Geng, Q., and Qiao, Z. 2016. Parameter tuning of particle swarm optimization by using taguchi method and its application to motor design. *2014 4th IEEE International Conference on Information Science and Technology (ICIST)*. Shenzen, Guandong, China. 26-28 April 2014. 722 – 726.

[38] Sadeghi, A., Alem-Tabriz, A., and Zandieh, M. 2011. Product portfolio planning: A metaheuristic-based simulated annealing algorithm. *International Journal of Production Research* 49(8): 2327-50.

[39] Candan, G., and Yazgan, H. R. 2015. Genetic algorithm parameter optimisation using taguchi method for a flexible manufacturing system scheduling problem. *International Journal of Production Research* 53(3): 897-915.

[40] Wu, H., Nie, C., Kuo, F.-C., Leung, H., and Colbourn, C. J. 2015. A discrete particle swarm optimization for covering array generation. *IEEE Transactions on Evolutionary Computation* 19 (4): 575 - 591.

[41] McConnell, J., Nunnally, B. K., and McGarvey, B. 2011. Meeting specifications is not good enough—the taguchi loss function. *Journal of Validation Technology*:38-42.

[42] García, S., Fernández, A., Luengo, J., and Herrera, F. 2009. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. Information Sciences 180:2044–64.

[43] Derrac, J., García, S., Molina, D., and Herrera, F. 2011. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation 1 (1):3-18.