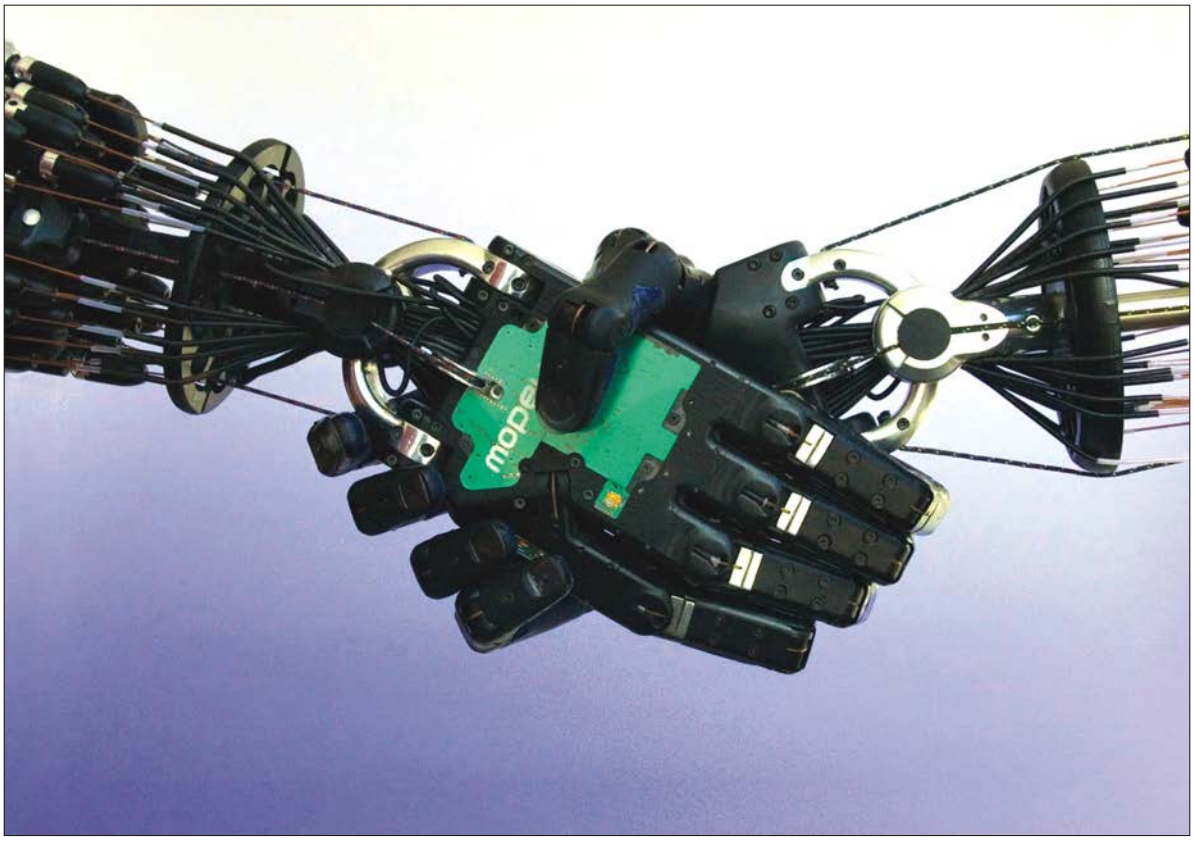

Journal of Artificial Intelligence

ISSN 1994-5450

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Artificial Intelligence 5 (2): 85-90, 2012
ISSN 1994-5450 / DOI: 10.3923/jai.2012.85.90

A Greedy Particle Swarm Optimization Strategy for T-way Software Testing

Bestoun S. Ahmed and Kamal Z. Zamli
Software Engineering Group, School of Electrical and Electronic Engineering, Universiti Sains Malaysia, 14300 Nibong Tebal, Penang, Malaysia

Corresponding Author: Bestoun S. Ahmed, Software Engineering Group, School of Electrical and Electronic Engineering, Universiti Sains Malaysia, 14300 Nibong Tebal, Penang, Malaysia

ABSTRACT

Combinatorial strategies are used as methods or mechanisms for selecting test cases using combinations of test input parameters. We normally want that all t-way combinations of parameter values occur in the test suit at least once. Artificial intelligence base search algorithms have been used within strategies for constructing near optimal test suites. In this paper, we propose a new test generation strategy, for combinatorial testing based on greedy Particle Swarm Optimization. The basic design concepts of the strategy are demonstrated through the paper. The experimental results and comparisons of our strategy showed impressive results as far as the test suite size is considered.

Key words: Combinatorial interaction testing, test generation strategies, t-way testing, search-based software testing, particle swarm optimization

INTRODUCTION

In the complex software systems, such as those used in industrial applications, there is a large number of possible test cases due to the different input parameters the system may accept. These parameters and their interactions with each other must be considered to ensure the accurate detection of different software bugs (Qu *et al*., 2007). However, exhaustive testing is often impossible due to time and resource limitations (Lei *et al*., 2008). As a result, there still exists a need for a more intelligent mechanism to ensure that such interaction coverage is approached systematically with a minimum number of test cases. For solving such a problem, combinatorial strategy used which is a method or mechanism for selecting test cases using a combination of test input parameters (Beizer, 1990).

Due to the complexity of search space in the combinatorial interaction problems, different techniques have been used to deal with this search process. Artificial intelligent techniques have been regarded as being especially adequate search strategies, since they are able to deal with search for optimization (Afzal *et al*., 2009). Two of the well known algorithms are Genetic Algorithm (GA) and Ant Colony Algorithm (ACA). However, other heuristic search techniques have started to compete with GA and ACA such as Particle Swarm Optimization (PSO) in the context of algorithm simplicity and performance (Wang *et al*., 2011; Marinakis and Marinaki, 2010a; Marinakis and Marinaki, 2010b; Pant *et al*., 2008). Recent literature showed that PSO outperforms GA and ACA in different cases of optimization problems (Windisch *et al*., 2007).

Motivating by those researches, this paper a new test suite generation strategy for t-way combinatorial testing (whereby, t indicates the interaction strength) based on Particle

Swarm Optimization, namely PSTG. PSTG complements our earlier work on pairwise testing (Ahmed and Zamli, 2011) to investigate the use of PSO for testing when t = 2. In addition to its impressive results against other strategies in case of test case size, PSTG generates one test case at a time.

Particle swarm optimization: Particle swarm optimization is a mechanism that tries to manipulate a certain number of candidate solutions at once (Chettih *et al*., 2011; Qasem and Shamsuddin, 2010). The whole population is called swarm and the solutions are called particles (Jie *et al*., 2008; Poli, 2008). Each solution represented by a particle that works in the search space to find a better position or solution of the problem. As a popular optimization method, PSO has been used during the last years since it showed a number of advantages in comparison to other optimization methods. As compared to other artificial intelligent optimization methods, PSO has few parameters to regulate and can be easily merged with the environment that needs optimization. In additions, PSO does not need the calculation of derivatives that the knowledge of good solutions is kept by all particles and that particle share the information with others in the swarm (Ganjali, 2008; Yap *et al*., 2011; Padhy, 2009; Sutha and Kamaraj, 2008).

With the starting of the optimization process in PSO, each particle has a random position and updates its position iteratively in the hope of finding better solutions. This is done with each particle by holding the essential information about its movement (Marinakis and Marinaki, 2010a; Marinaki *et al*., 2010). These information including its position currently (x_i), its velocity currently (v_i), personal best or the position that it has achieved so far which is denoted by ($pBest_i$) of particle i, local best or the position that it has achieved in its neighborhood which is denoted by ($lBest_i$) and the global best or the position it has achieved in the whole swarm which is denoted by ($gBest_i$). The manipulation of the particles around the search space is restricted by a certain update and positions rule as follows (Windisch *et al*., 2007; Ganjali, 2008):

$$V_{j,d}(t) = w\,V_{j,d}(t\text{-}1)+c\,r_{j,d}(pBest_{j,d}(t\text{-}1)-X_{j,d}(t\text{-}1))+c'r'_{j,d}+(lBest_{j,d}(t\text{-}1)-X_{j,d}(t\text{-}1)) \quad (1)$$

$$X_{j,d}=X_{j,d}(t\text{-}1)+V_{j,d}(t) \quad (2)$$

where, t is iteration number or time, d is the dimension, j the particle index, w is the inertia weight, r and r' are two random factors which are two random real numbers between 0 and 1 and c, c' are acceleration coefficients that are adjusting the weight between components. Pursuant to such updated rule, each particle update its velocity for better movement around the search space and the new velocity used to find the new position of the particles depending on a cost factor that controls this movement.

A strategy for t-way test suite generation: In test suite generation, we are mainly dealing with parameters and values and we want to find optimal test cases that cover most of the interaction elements. We introduce each particle as a vector. Since each test case has (D) parameters, as a result, the particle or the vector is (D) dimension also. We can illustrate this vector by the notation: $X_j = (X_{j,1}, X_{j,2}, X_{j,d}, \ldots\ldots\ldots, X_{j,D})$.

Referring to Fig. 1, our test suite generation strategy will start by receiving the parameters and values. The strategy immediately manipulates all parameters' values. Then, the algorithm will generate all t-way combinations named Ps that contains all t-interaction element combinations of parameters' values that are not been covered yet (step 4).

```
1: Input: Parameters' values, strength of coverage t;
2: Output: A test case;
3: Let Ps be a set of all not covered combinations of parameter values;
4: Generate Ps;
5: Let Ts be a set of candidate tests;
6: While Ps is not empty do {//termination-condition
7: Randomly initialize particles Xi(t) and velocities Vi(t);
8: For a specific number of iterations do {
9: Evaluate Xi(t) for its T-interaction element coverage with Ps;
10: if Xi(t) covered maximum interaction element in PS{
11: Add Xi(t) to final test suit Ts;
12: Remove Xi(t) from Ps;
13: continue;
14: }
15: else {
16: Choose best coverage particle to be lBest;
17: Calculate Vi(t+1) according to lBest;
18: Move Xi(t) to Xi(t+1) according to Vi(t+1);
19: }
20: Evaluate Xi(t+1);
21: If lBest(t+1) cover bigger T-interaction elements;
22: lBest=lBest(t+1);
23: }//End for
24: Let gBest be the best test case found;
25: gBest = lBest(t+1);
26: Add gBest to the test set Ts;
27: Remove those combinations in Ps that covered by;
28: }//End while
```

Fig. 1: A test case generation procedure for PSO

When a test case is found for Ts that can cover more t-interaction elements, the strategy removes the t-combinations which are covered by this test case, from Ps list. The strategy continues its running until Ps list get empty (step 6). The strategy randomly initializes each particle in the swarm search space with its associated parameter values (step 7). It compares each particle which represents a test case, with the list of t-interaction element PS (steps 9 and 20).

EXPERIMENTAL RESULTS

To justify and evaluate the efficiency of our strategy in term of the generated test suite size, we made comparison with some existing strategies and tools based on well-known benchmarks. These strategies are IPOG with its tool FireEye (Lei *et al.*, 2007), WHITCH, Jenny, TConfig and TVG. The comparison aims to study the growth in the generated test suite size in terms of strength of coverage (t). We adopt two different set of experiment conducted by Lei *et al.* (2007) and Bryce *et al.* (2005).

All strategies are employed within our environment which consisted of a desktop PC with Windows XP, 2.8 GHz Core 2 Due CPU, 2 GB of RAM and JDK 1.5 installed.

Table 1 and 2 showed the results obtained for the two set of experiments. Each table represents the smallest test suit size obtained. The cells were marked NS (not supported) indicate that the tool cannot generate the test case for a specific configuration and the cells were marked NA (not available) indicate that the results were unavailable.

Referring to the above tables, we note that our PSTG strategy scales well against other strategies in most cases. Referring to Table 1, PSTG produces optimal sizes in case of t equals to 2 and 4 while in case of t equals to 3 and 5 it can compete the other strategies except the optimal one.

Table 1: Comparison with existing algorithms

Configurations	Density	Para order	TVG	PICT	AETG	mAETG	GA	ACA	GA-N	IPO-N	IPO	IPOG	Jenny	PSTG
CA (N;2, 3^4)	NA	NA	12	NA	9	9	9	9	NA	NA	9	12	13	9
CA (N;2, 3^{13})	NA	NA	20	NA	15	17	17	17	NA	NA	17	12	20	17
MCA (N;2, 6^1 5^1 4^6 3^8 2^3)	NA	NA	41	NA	34	35	33	32	NA	NA	NA	36	31	39
MCA (N;2, 7^1 6^1 5^1 4^6 3^8 2^3)	NA	NA	52	NA	45	44	42	42	NA	NA	NA	44	51	49
MCA (N;2, 5^1 3^8 2^2)	NA	NA	23	NA	19	20	15	16	NA	NA	NA	19	41	21
CA (N;3, 3^6)	53	53	48	48	45	38	33	52	52	47	28	53	51	42
CA (N;3, 4^6)	64	106	120	111	105	77	64	64	85	64	64	64	112	102
CA (N;3, 5^6)	213	225	239	215	NA	194	125	125	223	173	200	216	215	220
CA (N;3, 6^6)	362	363	409	369	343	330	331	330	389	271	366	383	373	338
CA (N;3, 5^7)	242	225	269	241	229	218	218	218	336	199	239	274	236	229
MCA (N;3, 10^1 6^2 4^3 3^1)	365	379	429	368	NA	377	360	361	373	368	464	361	397	385

NS: Not supported, NA: Not available

Table 2: P and V constants (10, 2) but t varied up to 6

t-way	IPOG	Whitch	Jenny	T config	TVG	PSTG
2	50	45	45	48	50	45
3	313	225	290	312	342	300
4	1965	1750	1719	1878	1971	1716
5	11009	NS	9437	NA	NA	9752
6	57290	NS	NA	NA	NA	56113

NS: Not supported, NA: Not available

WHITCH appeared to produce satisfactory results in case of small value of t (maximum of 4) and not producing results beyond that. Similarly, TConfig and TVG do not produce any specific results for more than one day of running in case of t>4. Jenny produce a reasonable result in case of t equals to 5 while it cannot produce results in case of t equals to 6.

With the multiple domain (variable) configurations in Table 2, also PSTG scales well against others in most cases. In most of the tests PSTG can produce the most optimum; however, when it is not the most optimum in some cases, it still can compete with most of the other none optimum strategies.

CONCLUSION

We have proposed and illustrated our efficient strategy, namely, PSTG for t-way combinatorial test case generation using a novel approach by combining the greedy fashion of particle swarm optimization technique with the software test case generation to gain near optimal solution. The main concern of our algorithm is optimization in term of size of the resulting test sets. From the experiment results, we can state that no single strategy can claim absolute dominance over other strategies for all configuration since it is an NP-complete problem (Lei *et al*., 2007; Lei and Tai, 1998). PSTG performs better than other strategies in term of size in most cases but it cannot be the most optimal strategy for all configuration sets. We are currently developing our PSTG strategy to release the beta version of the PSTG tool.

ACKNOWLEDGMENT

This research is partially funded by the generous fundamental grants-"Investigating t-way Test Data Reduction Strategy Using Particle Swarm Optimization Technique" from Ministry of Higher Education (MOHE) and the USM Research University grants-"Development of variable-strength interaction testing strategy for t-way Test Data generation" and the short term grant-

"Development of interaction testing tool for pairwise coverage with seeding and constraints". The first author, Bestoun S. Ahmed, is a recipient of the USM fellowship.

REFERENCES

Afzal, W., R. Torkar and R. Feldt, 2009. A systematic review of search-based testing for non-functional system properties. Inform. Software Technol., 51: 957-976.

Ahmed, B.S. and K.Z. Zamli, 2011. The development of a particle swarm based optimization strategy for pairwise testing. J. Artif. Intell., 4: 156-165.

Beizer, B., 1990. Software Testing Techniques. 2nd Edn., Van Nostrand Reinhold, New York, ISBN: 0-442-20672-0, Pages: 550.

Bryce, R., C.J. Colbourn and M.B. Cohen, 2005. A framework of greedy methods for constructing interaction tests. Proceeding of the 27th International Conference on Software Engineering, May15-21, 2005, ACM Press, St. Louis, MO., USA., pp: 146-155.

Chettih, S., M. Khiat and A. Chaker, 2011. Voltage control and reactive power optimisation using the meta heuristics method: Application in the Western algerian transmission system. J. Artif. Intell., 4: 12-20.

Ganjali, A., 2008. A requirements-based partition testing framework using particle swarm optimization technique. Master Thesis, Electrical and Computer Engineering, University of Waterloo, Ontario, Canada.

Jie, J., J. Zeng, C. Han and Q. Wang, 2008. Knowledge-based cooperative particle swarm optimization. Applied Math. Comput., 205: 861-873.

Lei, Y. and K.C. Tai, 1998. In-parameter-order: A test generation strategy for pairwise testing. Proceedings of the 3rd IEEE International Symposium on High-Assurance Systems Engineering, November 13-14, 1998, Washington DC. USA., pp: 254-261.

Lei, Y., R. Kacker, D.R. Kuhn, V. Okun and J. Lawrence, 2007. IPOG: A general strategy for T-way software testing. Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, March 26-29, 2007, IEEE Computer Society, Tucson, AZ., USA., pp: 549-556.

Lei, Y., R. Kacker, D.R. Kuhn, V. Okun and J. Lawrence, 2008. IPOG/IPOG-D: Efficient test generation for multi-way combinatorial testing. Softw. Test. Verification Reliab., 18: 125-148.

Marinakis, Y. and M. Marinaki, 2010a. A hybrid genetic: Particle swarm optimization algorithm for the vehicle routing problem. Exp. Syst. Appl., 37: 1446-1455.

Marinakis, Y. and M. Marinaki, 2010b. A hybrid multi-swarm particle swarm optimization algorithm for the probabilistic traveling salesman problem. Comput. Operations Res., 37: 432-442.

Marinaki, M., Y. Marinakis and G.E. Stavroulakis, 2010. Fuzzy control optimized by PSO for vibration suppression of beams. Control Eng. Pract., 18: 618-629.

Padhy, N.P., 2009. Artificial Intelligence and Intellegent Systems. Oxford University Press, Oxford.

Pant, M., P. Sharma, T. Radha, R.S. Sangwan and U. Roy, 2008. Nonlinear optimization of enzyme kinetic parameters. J. Boil. Sci., 8: 1322-1327.

Poli, R., 2008. Analysis of the publications on the applications of particle swarm optimisation. J. Artif. Evol. Appl., Vol. 2008, 10.1155/2008/685175

Qasem, S.N. and S.M. Shamsuddin, 2010. Generalization improvement of radial basis function network based on multi-objective particle swarm optimization. J. Artif. Intell., 3: 1-16.

Qu, X., M.B. Chohen and K.M. Woolf, 2007. Combinatorial interaction regression testing: A study of test case generation and prioritization. Proceedings of the IEEE International Conference on Software Maintenance, October 2-5, 2007, University of Nebraska-Lincoln, Lincoln, pp: 255-264.

Sutha, S. and N. Kamaraj, 2008. Particle swarm optimization applications to static security enhancement using multi type facts devices. J. Artif. Intell., 1: 34-43.

Wang, J., Y. Cai, Y. Zhou, R. Wang and C. Li, 2011. Discrete particle swarm optimization based on estimation of distribution for terminal assignment problems. Comput. Ind. Eng., 60: 566-575.

Windisch, A., S. Wappler and J. Wegener, 2007. Applying particle swarm optimization to software testing. Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, July 7-11, 2007, ACM Press, London, England, pp: 1121-1128.

Yap, D.F.W., S.P. Koh, S.K. Tiong and S.K. Prajindra, 2011. Particle swarm based artificial immune system for multimodal function optimization and engineering application problem. Trends Applied Sci. Res., 65: 282-293.