2010 34th Annual IEEE Computer Software and Applications Conference Workshops

An Optimization Strategy for Evolutionary Testing Based on Cataclysm*

Meng Wang[1], Bixin Li[1], Zhengshan Wang[1,2], Xiaoyuan Xie[1]
[1]School of Computer Science and Engineering, South University, Nanjing, China
[2]Department of Computer Science and Technology, Chuzhou University, Chuzhou, China

***Abstract*—Evolutionary Testing (ET) is an effective test case generation technique which uses some meta-heuristic search algorithm, especially genetic algorithm, to generate test cases automatically. However, the prematurity of the population may decrease the performance of ET. To solve this problem, this paper presents a novel optimization strategy based on cataclysm. It monitors the diversity of population during the evolution process of ET. Once the prematurity is detected, it will use the operator, cataclysm, to recover the diversity of the population. The experimental results show that the proposed strategy can improve the performance of ET evidently.**

***Keywords*-Evolutionary Testing; premature; cataclysm; diversity measure;**

I. Introduction

Software testing is an efficient and expensive part of software development process. It consumes more than 50% of the overall development effort and budget [2]. And the design of test cases is essential to assure the test quality. So it has been desirable to automate the test case generation. In structural testing and functional testing, test cases are always generated manually according to the specified standards, which is difficult to be automated. The random testing, even though, which has a high automation, may generate excessive test cases that have low possibility to satisfy the testing requirement and low efficiency to detect bugs in software [3].

Evolutionary Testing (ET) is a promising automation technique, which can generate test cases automatically for different test objectives according to certain test requirements. ET designates the use of a kind of meta-heuristic search method, especially Genetic Algorithm (GA), for the test case generation, which converts the generation of test cases into an optimal problem [1,2,4,10]. And different configuration strategies may greatly affect the performance of ET. As a result, it is valuable to find a proper configuration strategy.

However even if it has a best static configuration strategy, ET cannot promise to find the global optimal solution. During the evolution process, the population of ET may get trapped into a local optimal solution, which means the evolution of ET is premature. However ET cannot get rid of prematurity itself. And finally ET can only find a local optimal solution, which is not the one we need.

This paper presents a kind of ***O***ptimization ***S***trategy ***B***ased on ***C***ataclysm (***OSBC***) for structural evolutionary testing. OSBC monitors the diversity of the population during the evolution process, detects the prematurity by calculating the diversity and comparing it with a threshold level and uses cataclysm to recover the diversity of the population. The empirical results show that OSBC can greatly improve the performance of the ET.

The rest of the paper is organized as follows. Section 2 briefly introduces the premature problem of ET. In section 3, the new optimization strategy OSBC is described in details. Empirical results and analysis are presented in section 4. Finally, section 5 presents the conclusions and our future work.

II. Premature Problem of Evolutionary Testing

ET designates the use of a kind of meta-heuristic search method, especially Genetic Algorithm (GA), for the test case generation. The input domain of the test object forms the search space in which ET searches for a test case that fulfills the respective test goal [4]. ET consists of code strategies, fitness function, genetic operations and so on. Some common code strategies and genetic operations are shown in Table I [10].

Table I: Basic methods of Evolutionary Testing

Parameter	Method	Description
Code	Binary	Encode individuals by binary code
	Gray	Encode individuals by Gray code
Selection	Random	Select pairs of individuals randomly
	Spin	Select pairs of individuals based on the spin strategy
Crossover	Single	Intercross a pair of genes in a single cross point
	Uniform	Intercross a pair of genes in several cross points
Mutation	Bit	Alter a bit value of a gene randomly
Survival	Fitness	Reinsert genes with the higher fitness values
	Unrepeatable	Reinsert unique genes with higher fitness values
	Spin	Reinsert genes based on spin strategy

*This work is supported partially by National High Technology Research and Development Program under Grant No. 2007AA01Z141 and No. 2008AA01Z113, partially by the National Natural Science Foundation of China under Grant No. 60773105 and No. 60973149, and partially by the Natural Science Foundation of Chuzhou University under Grant No.2008KJ003B.

Correspondence to: Bixin Li, School of Computer Science and Engineering, Southeast University, Nanjing, China. E-mail: bx.li@seu.edu.cn

In ET, an individual in the population represents a test input, and the fitness function is designed according to the specific test target [4]. The goal of ET is to find a test case, which is the global optimal solution, that satisfies the test requirement. In structural tesing, the test requirement is usually a coverage standard (path coverage, branch coverage, etc.).

As is known, GA itself cannot guarantee to find the global optimal solution. GA may get trapped into a local optimal solution, which means the population of GA is premature. A population of GA is said to be premature, when the best individual of the population does not improve with the evolution of GA or most individuals of the population are similar to or the same as each other [9]. When prematurity occurs, GA itself cannot get rid of the prematurity and the evolution of GA will lose its power of finding the global optimal solution. And finally a local optimal solution is found, which is not the global optimal solution.

Prematurity also occurs in the evolution of ET. In traditional ET, proper configuration strategies may help to prevent the prematurity in some cases [4,5,6,7,10,11,12]. However even with the best configuration strategy, ET cannot promise to find the global optimal solution. And when prematurity occurs, few strategies can help ET get rid of prematurity.

Since ET cannot promise to find the global optimal solution, lots of attention has been paid to the performance improvement of ET. H. Sthamer offers an idea that generates test cases with GA based on Control Flow Graph [1]. Andre Baresel offers a technology that designs the fitness function to improve the performance of ET [4]. Mark Harman applies program slicing to dealing with the flag problem in ET [5]. Mark Harman also applies program transformation to improve testability [6,7]. Liang Shi gives some static configuration strategies to improve the performance of ET [12]. Although they can improve the performance of ET, the ideas mentioned above can only help prevent the occurrence of prematurity but fail to help ET get rid of prematurity.

Xiaoyuan Xie offers DOMP in [10]. Xie focuses on the dynamic optimization of mutation possibility (DOMP) to recover the diversity of the population. DOMP monitors the evolution of ET dynamically. Once DOMP detects the prematurity, the mutation possibility is increased to a high level so that the prematurity can be broken. In the following evolution of ET, once the diversity of the population is recovered, the mutation possibility drops to the ordinary level [10]. However in some cases, DOMP does not work well. And Xie also offers two ideas of Annealing Genetic Algorithm (AGA) and Restricted Genetic Algorithm (RGA) in [11] to deal with the population prematurity.

III. Optimization Strategy Based on Cataclysm

From the above discussion, it can be figured out that the prematurity of ET is a result of the loss of the diversity of the population. Some configuration strategies with low selection pressure may be chosen to prevent the occurrence of prematurity [10]. However they may slow down the evolution speed and cannot recover the diversity when the evolution is premature. Currently, DOMP can recover the diversity of the population when prematurity occurs. But with certain configuration strategies, DOMP does not work very well.

Some self-adaptive strategies may be introduced to deal with the prematurity of the population. It is possible for us to monitor the diversity of the population to judge whether the evolution of ET is in a healthy status. If the diversity of the population is too high, the individuals of the population are too separated to converge, which makes ET become random testing. On the contrary, if the diversity of the population is too low and ET has not found the global optimal solution, the evolution of ET has gotten trapped into a local optimal solution and the global optimal solution cannot be found in the following evolution of ET. In both cases, the performance of ET decreases. As a result, during the evolution of ET, we may monitor the diversity of the population and make it within a proper range so that we may find a global optimal solution.

The optimization strategy presented in this paper is denoted as $\boldsymbol{OSBC}$ ($\boldsymbol{O}$ptimization $\boldsymbol{S}$trategy $\boldsymbol{B}$ased on $\boldsymbol{C}$ataclysm). The basic idea of OSBC simulates the cataclysm in the nature. When cataclysm occurs in the nature, most lives of the population are killed. Only the best individuals can survive. And after the cataclysm, these survived individuals live and multiply and soon a new population turns up. The best individual of the new population is not worse than that of the old population.

Cataclysm can also be used in ET when prematurity occurs. During the evolution process of ET, OSBC monitors the diversity of the current population based on certain measure methods. If the diversity is lower than some threshold level T_{low} and the global optimal solution has not been found, it can be judged that prematurity occurs. Then the genetic operator, cataclysm, begins to work. The exact steps of cataclysm are as follows: 1) the best individual in the premature population is kept down to the new population; 2) the other individuals of the premature population are killed and then generated randomly for the new population. After the new population is generated, the diversity of the population is recovered and the evolution of ET can become normal, which means the evolution of ET has gotten rid of prematurity.

Besides, some other operations can be added into the evolution process of ET. Firstly, if the diversity of the initial population is higher than a threshold level T_{high}, it will take a long time to find the global optimal solution or even worse fail to converge. So at the initialization of the population, if the diversity of the population is higher than T_{high}, generate the initial population again until the diversity of the initial

Table II: Different functions in experiments

Function name	Input	Condition to satisfy (testing objective)	Corresponding result	Description of the function
Triangle	a,b,c	(a==b)&&(b==c)	The triangle is an equilateral one.	a,b and c are the edges of a triangle.The program uses them to classify the triangle.
Quadratic	a,b,c	$b^2 - 4 * a * c == 0$	The equation has two equal real solutions.	a, b and c are the coefficients of a quadratic equation. The discriminant $b^2 - 4 * a * c$ is used to investigate the distribution of the solutions.
LineCover	l1x,l2x, l1y,l2y, rx,ry, w,h	(rx==min(l1x,l2x)) &&(ry==min(l1y,l2y)) &&(rx+w==max(l1x,l2x)) &&(ry+h==max(l1y,l2y))	The line segment is the diagonal of the rectangle.	(l1x,l1y) and (l2x,l2y) are 2 end points of a line segment.(rx,ry) is the bottom left point of a rectangle. (w,h) are the width and height of a rectangle. The program checks the position relationship between the line segment and the rectangle.

population is less than T_{high}. Secondly, when the survive operator runs, the best individual of the parent population may be kept down to the offspring population so that it can make sure that the offspring population is at least not worse than the parent population or even is much better.

OSBC has four parameters: Observation Frequency of OSBC (OFO), Diversity of the Current Population (DCP), the higher threshold level of the diversity (T_{high}) and the lower threshold level of the diversity (T_{low}). At the initialization of the population, we need make sure that DCP is less than T_{high}. During the evolution of ET, OSBC monitors the population each OFO generations. In once observation, OSBC measures DCP. If DCP is less than T_{low}, prematurity occurs and cataclysm is taken. The configuration strategy of the parameters is very important in ET. Proper configuration strategy can improve the performance of ET greatly.

OFO reflects the frequency that OSBC monitors the population. It can be assigned with a value between 1 and the maximum of the iteration. The smaller OFO is, the higher the frequency is. If OFO is too small, it will cost a lot of time to monitor the population and the performance decreases. Since the diversity of ET cannot change greatly in a few generations, we may set OFO a bit higher. In our experiment, we set OFO=10, so that OFO can detect prematurity without too much cost of time.

DCP reflects the diversity of the current population. We take the entropy diversity measure method presented in [9].

$$DCP = -\sum P_k * log P_k$$

A partition P_k is the proportion of the k class fitness value in the population. We can monitor the diversity of the population according to the formula mentioned above. If DCP is very high, it means the distribution of the individuals is separated. If DCP is too low, it means lots of individuals are similar or even the same.

T_{high} and T_{low} reflect the threshold level which offers the judgment standard. And the valid value of T_{high} and T_{low} is between 0 and 2.0 according to the formula which calculates DCP. If DCP is higher than T_{high}, the individuals of the initial population are too separated to converge and the initial population need be regenerated. If DCP is less than T_{low} and ET has not found the global optimal solution, prematurity occurs and cataclysm need be taken. T_{high} and T_{low} is critical to OSBC. If T_{high} and T_{low} do not have proper values, the performance of ET may greatly decrease.

IV. Experimental Study

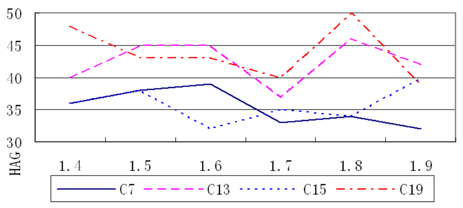
In this section, we will evaluate the performance of our proposed strategy OSBC by comparing it with existing DOMP. For some large and complex programs, it is a bit difficult to trigger the occurrence of prematurity. As a result, in our experiments, in order to trigger the prematurity of ET easily, we select three simple programs, which do not contain any loop conditions, as the test targets(shown in Table II). The test criterion of experiments is to cover a particular branch of the test targets(Branch Coverage).

A. Parameter Settings

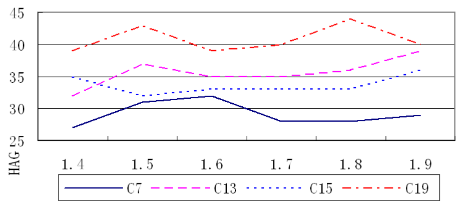
For each configuration strategy of ET, suppose its total number of experiments is T_0, the size of population is P_{size}, and the maximum iteration in an experiment is IterMax. And in order to compare the performance of OSBC with DOMP, we take the performance parameters used in [10], as follows:

- Hit Percent (P_h): P_h = T_h / T_0, where T_h is the number of experiments that achieves the testing objective. And P_h is an essential index to evaluate the performance of ET. The higher P_h is, the more effective ET is.
- Hit Average Generation (HAG): the average number of the generations of experiments that achieves the testing objective. HAG reflects the converge speed of ET. The lower HAG is, the faster ET is.
- Total Test Cases (TTC): the average number of test cases generated in experiments.
  TTC = P_{size}*[P_h * HAG + (1- P_h)* IterMax]
  The lower TTC is, the lighter ET is.

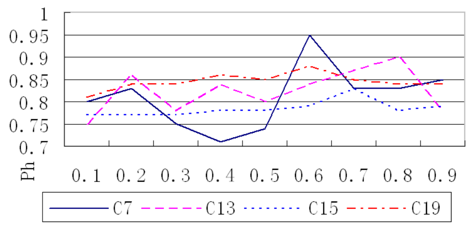
In our experiments, the parameters of ET are as follows: the crossover possibility P_c(assigned with 0.9), the mutation possibility P_m (assigned with 0.1), the survive possibility P_s (assigned with 0.5), the size of the population P_{size} (assigned with 100), the max iteration IterMax (assigned with 250) and the total number of experiments T_0 (assigned with 300).



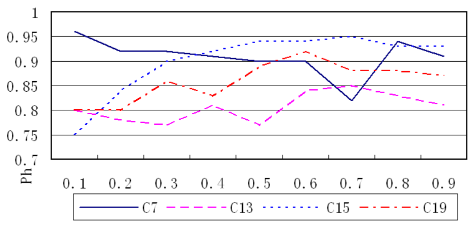
(a) HAG of Triangle



(b) HAG of Quadratic



(c) P_h of Triangle



(d) P_h of Quadratic

Figure 1: Parameter search for T_{high} and T_{low}

Table III: Configuration strategies of ET

	Code		Selection		Crossover		Survival		
	B	G	R	S1	S2	U1	S3	F	U2
C1	*		*		*		*		
C2	*		*		*			*	
C3	*		*		*				*
C4		*	*		*		*		
C5		*	*		*			*	
C6		*	*		*				*
C7	*		*			*	*		
C8	*		*			*		*	
C9	*		*			*			*
C10		*	*			*	*		
C11		*	*			*		*	
C12		*	*			*			*
C13	*			*	*		*		
C14	*			*	*			*	
C15	*			*	*				*
C16		*		*	*		*		
C17		*		*	*			*	
C18		*		*	*				*
C19	*			*		*	*		
C20	*			*		*		*	
C21	*			*		*			*
C22		*		*		*	*		
C23		*		*		*		*	
C24		*		*		*			*

B: Binary Code, G: Gray Code, R: Random Selection, S1: Spin Selection, S2: Single Crossover, U1: Uniform Crossover, S3: Spin Survival, F: Fitness Survival, U2: Unrepeatable Survival

It is impossible for us to know the compatible parameter setting of ET in advance. So we conducte a series of experiments to investigate the optimal parameter setting of OSBC by testing the objects listed in Table II. The experiment results show that the performance of ET varies with the change of T_{high} and T_{low} which are critical parameters of OSBC.

In LineCover, OSBC does not affect the performance of ET evidently. So we give the result of Rectangle and Quadratic. The configuration strategies we choose are C7, C13, C15 and C19 (listed in Table III). After a lot of experiments, we find that T_{high} does not affect P_h evidently. So we choose the performance parameter HAG. The variation of HAG with the change of T_{high} can be seen in subgraph (a) and (b) of Figure.1. It can be found that (T_{high}=1.7) outperforms others on average. But the change of T_{low} affects the variation of P_h greatly. So we choose the performance parameter P_h. Subgraph (c) and (d) of Figure.1 show the variation of P_h with the change of T_{low}. It can be found that (T_{low}=0.6) outperforms others on average. Consequently the parameter configuration strategy (T_{high} =1.7, T_{low}=0.6) outperforms other parameter configuration strategies.

B. *Performance Improvement with OSBC*

To examine the validity of OSBC, we compare the performance among ET without optimization, ET with DOMP and ET with OSBC, by testing the three test targets based on 24 configuration strategies listed in Table III [10].

Table IV shows the performance comparison among ET without optimization(Un_O), with DOMP and with OSBC in Triangle. It is observed that P_h is increased obviously by OSBC in most cases. It can be learnt from Table IV that there is no configuration strategy in which P_h is decreased by OSBC compared with ET without optimization. Firstly

Table IV: Performance comparison among Un_O, DOMP and OSBC in Triangle

	P_h			HAG			TTC		
	U	D	O	U	D	O	U	D	O
C1	0.78	0.90	0.92	18	30	20	6904	5046	3840
C2	1.00	1.00	1.00	31	31	31	3100	3100	3100
C3	0.98	0.99	1.00	15	17	15	1970	1933	1500
C4	1.00	1.00	1.00	5	5	4	500	500	400
C5	1.00	1.00	1.00	25	27	25	2500	2700	2500
C6	1.00	1.00	1.00	7	7	5	700	700	500
C7	0.68	0.87	0.95	28	35	33	9748	6295	4375
C8	1.00	1.00	1.00	36	34	38	3600	3400	3800
C9	0.91	0.93	1.00	26	21	24	4548	3703	2400
C10	1.00	1.00	1.00	5	6	5	500	600	500
C11	1.00	1.00	1.00	27	26	25	2700	2600	2500
C12	1.00	1.00	1.00	10	9	7	1000	900	700
C13	0.74	0.87	0.84	24	35	37	8124	6295	7108
C14	0.75	0.79	0.84	16	28	30	7379	7462	6520
C15	0.76	0.81	0.79	28	30	35	7972	7114	8015
C16	1.00	1.00	1.00	3	3	2	300	300	200
C17	1.00	1.00	1.00	4	3	3	400	300	300
C18	1.00	1.00	1.00	4	3	3	400	300	300
C19	0.62	0.84	0.88	24	37	40	10988	7108	6520
C20	0.60	0.83	0.80	14	27	35	10681	6491	7800
C21	0.66	0.77	0.90	26	30	29	10059	7994	5110
C22	1.00	1.00	1.00	3	4	2	300	400	200
C23	1.00	1.00	1.00	4	5	4	400	500	400
C24	1.00	1.00	1.00	4	4	3	400	400	300

U: ET without optimization, D: DOMP, O: OSBC

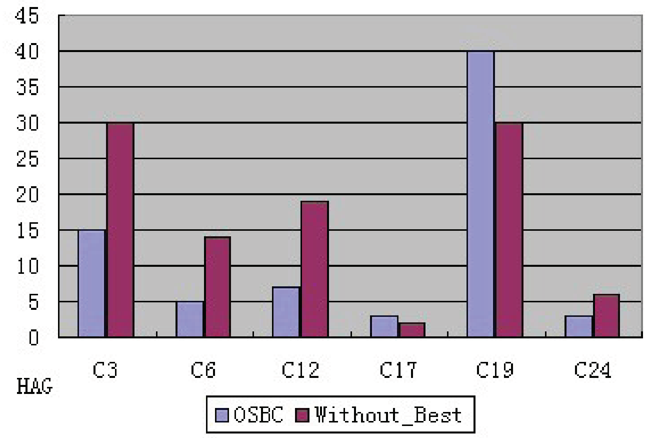


Figure 2: the comparison of OSBC and Without_Best

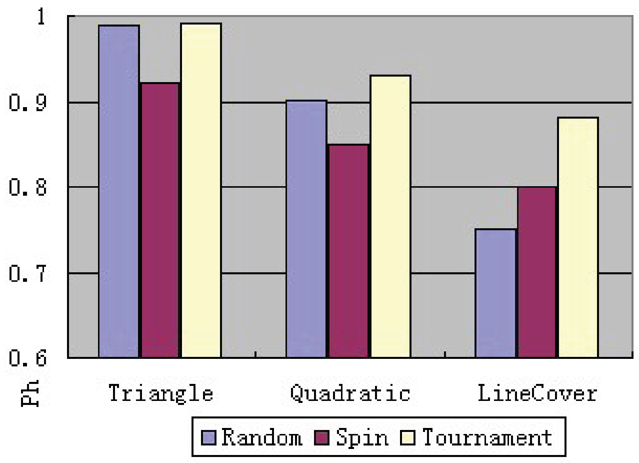


Figure 3: Comparison between Random, Spin and Tournament Selection

in P_h, except the configuration strategies (C13, C15 and C20), the other configuration strategies of OSBC work better than DOMP. And OSBC makes two more 100% hit (C3 and C9). Secondly in HAG, except the configuration strategies (C1, C7, C8, C13, C14, C15, C19, C20 and C21), the other configuration strategies of OSBC work better than DOMP. Thirdly in TTC, except the configuration strategies (C8 and C15), all the other configuration strategies of OSBC improve much more than DOMP or the same as DOMP.

It can be figured out that to find the global optimal solution, OSBC works better than DOMP. But the time cost of OSBC may be greater than DOMP, because OSBC almost begins a completely new search.

OSBC does not work well with LineCover just like DOMP. And the reason is that it is difficult for LineCover to get trapped into a local optimal solution.

During the evolution of ET, once prematurity occurs, cataclysm works. It can be considered that the search space of ET is enlarged. OSBC throws away the old search space and keep the best individual to the new search space. The best individual can help speed up the evolution process. So P_h, HAG and TTC are improved a lot.

In order to verify the use of the best individual of the premature population, we do some other experiments. When cataclysm works, if OSBC does not keep the best individual of the premature population down to the new population, we denote that as Without_Best. After a lot of experiments, we find out that the there is no much difference in P_h between Without_Best and OSBC . But the difference in HAG and TTC is greatly obvious between Without_Best and OSBC. We compare the HAG of the configuration strategies (C3, C6, C12, C17, C19 and C24), which are 100% hit, between OSBC and Without_Best. The result can be seen in Figure.2. It can be seen from Figure.2 that the best individual of the current population can help the new population to find the global optimal solution much quicker in most cases.

Besides, we also recommend another selection method (Tournament Selection). Figure.3 shows the comparison of Ph in Triangle among Random Selection, Spin Selection and Tournament Selection. It can be seen from the figure that the selection method(Tournament Selection) can improve the performance of OSBC greater than both Random Selection and Spin Selection on average.

The reason is that Tournament Selection stresses on the relative fitness value. After the cataclysm, the local optimal solution is kept down to the new population. And after crossing with other individuals, it enhances the possibility of getting rid of the local optimal solution so that ET can find other better individual and finally find the global optimal solution.

V. Conclusion and Future Works

In this paper, we present the idea of OSBC to improve the performance of ET, which can help ET get rid of prematurity. And empirical results show that the optimization strategy can greatly improve the performance of ET. And in some cases, OSBC works better than DOMP. But for the complicated branch conditions, OSBC does not work well.

There are some cases that OSBC does not work well in. If the branch condition is too complicate, OSBC loses its power. And with some configuration strategies, the time cost of OSBC is very high. So in future works, we will continue the research on the improvement of the performance of ET. And we plan to extend OSBC to more complicated problems, such as programs with loop conditions and so on. Since ET has so many configuration strategies, we may make OSBC itself find the fittest configure strategy according to a specific problem.

References

[1] H. Sthamer, ”The Automatic Generation of Software Test Data Using Genetic Algorithms”, PhD Thesis, University of Glamorgan, Pontypridd, Great Britain, April 1996.

[2] J. Wegener, A. Baresel, and H. Sthamer, ”Evolutionary Test Environment for Automatic Structural Testing”, Information and Software Technology Special Issue on Software Eng. Using Metaheuristic Innovative Algorithms, vol. 43, no. 14,pp. 841-854, 2001.

[3] J. Wegener, A. Baresel and H. Sthamer, ”Suitability of Evolutionary Algorithms for Evolutionary Testing”, in Proc. of 26th Annual International Computer Software and Applications Conference, Oxford, Great Britain, pp.287-289, 2002.

[4] A. Baresel, H. Sthamer and M. Schmidt, ”Fitness Function Design To Improve Evolutionary Structural Testing”, in Proc. of the Genetic and Evolutionary Computation Conference, New York, USA, 2002.

[5] M. Harman, L. Hu, R. Hierons, C. Fox, S. Danicic, J.Wegener, H. Sthamer, A. Baresel, ”Evolutionary Testing Supported by Slicing and Transformation ”, in The International Conference on Software Maintenance, Montreal, Canada, 2002.

[6] M. Harman,” Testability Transformation for Search-Based Testing”, Keynote of the 1st International Workshop on Search-Based Software Testing, 2008

[7] Hamilton Gross, Peter M. Kruse, Dr. Joachim Wegener, Dr. Tanja Vos” Evolutionary white-box software test with the EvoTest Framework , a progress report”, Proceedings of the IEEE International Conference on Software Testing, Verification, and Validation workshops , pp.111-120, 2009

[8] L. Kiran, M. Phil and M. Harmen,” Automated Test Data Generation for Coverage: Haven’t We Solved This Problem Yet?”, Proceedings of Testing: Academia Industry Conference Practice And Research Techniques (TAIC-PART ’09), 2009

[9] Xiaojin Wu and Zhongying Zhu, ”Research on Diversity Measure of Genetic Algorithms”, Information and Control, Vol,34, No.4, 2005.8

[10] Xiaoyuan Xie, Baowen Xu, Liang Shi, Changhai Nie and Yanxiang He,” A Dynamic Optimization Strategy for Evolutionary Testing”, in Pro. of the 12th Asia-Pacific Software Engineering Conference, TaiPei, Taiwan, 2005

[11] Xiaoyuan Xie, Baowen Xu and Changhai Nie,”Configuration Strategies for Evolutionary Testing”. 29th Annual International Computer Software and Applications Conference. Edinburgh, Scotland. July, 2005

[12] Liang Shi, Baowen Xu and Xiaoyuan Xie,”An Empirical Study of Configuration Strategies of Evolutionary Testing”, International Journal of Computer Science & Network Security IJCSNS, pp.44-49, 2006