A Dynamic Optimization Strategy for Evolutionary Testing[*]

Xiaoyuan Xie[1] Baowen Xu[1, 2, 3] Liang Shi[1] Changhai Nie[1] Yanxiang He[2]

[1] *Southeast University, Department of Computer Science & Engineering, 210096 Nanjing, China*
[2] *Key Laboratory of Software Engineering, Wuhan University, 430072 Wuhan, China*
[3] *National University of Defense Technology, Department of Computer, Changsha 410073, China*

Abstract

Evolutionary Testing (ET) is an efficient technique of automated test case generation. ET uses a kind of meta-heuristic search technique, Genetic Algorithm (GA), to convert the task of test case generation into an optimal problem. The configuration strategies of GA will have notable influences upon the performance of ET. In this paper, we present a dynamic self-adaptation strategy for evolutionary structural testing. It monitors evolution process dynamically, detects the symptom of prematurity by analyzing the population, and adjusts the mutation possibility to recover the diversity of the population. The empirical results show that the strategy can greatly improve the performance of the ET in many cases. Besides, some valuable advices are provided for the configuration strategies of ET by the empirical study.

Keywords: Software testing, evolutionary testing, structural testing, dynamic optimization

1. Introduction

Software testing is an efficient and expensive part of software development process. Usually, it consumes about 30~50% of the overall development effort and budget [3]. The systematic design of test cases is essential to assure the test quality. 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, has a high automaticity, may generate excessive test cases that have low possibility to satisfy the testing requirements and low efficiency to detect bugs in software [9].

Evolutionary Testing (ET) is a promising automation technique, which can generate feasible test cases automatically for various test objectives. [4, 7, 8, 14]

ET uses a kind of meta-heuristic search technique, Genetic Algorithm (GA), to convert the task of test case generation into an optimal problem. Therefore the configuration of GA may greatly affect the performance of search. It is valuable to provide some basic advices about the static configuration strategies of ET.

However, even if there was a best static configuration strategy for a program, it could not promise to find the optimal solutions. During the process of evolution, the population sometimes may prematurely converge on one local optimal solution, which badly delays the progress of evolution. And there is a lack of self-adaptation methods to avoid the delay.

In this paper, we present one kind of dynamic optimization strategies (DOMP) for evolutionary structural testing. It monitors evolution process dynamically, detects the symptom of prematurity by analyzing the population, and adjusts the mutation possibility to recover the population diversity. The empirical results show that the strategy can greatly improve the performance of the ET in many cases. Besides, we also provide some advices for the configuration strategies of ET by the empirical study. These advices are helpful to find specialized configurations for different programs.

The rest of the paper is organized as follows. Section 2 briefly introduces the performance problems of ET. In section 3, the new strategy DOMP is described in details. Empirical results and analysis are presented in section 4. Section 5 introduces the related works. And Section 6 presents the conclusions and our future works.

2. Performance problems of evolutionary testing

ET is based on the heuristic search technique GA that mimics the principle of biological evolution to seek the optimal solution for a complex problem. Some candidate solutions are encoded into genes by some *code strategies* to form the original population. Each

[*] This work was supported in part by the National Natural Science Foundation of China (60425206, 60373066, 60403016), National Grand Fundamental Research 973 Program of China (2002CB312000), and National Research Foundation for the Doctoral Program of Higher Education of China (20020286004).

Correspondence to: Baowen Xu, Department of Computer Science and Engineering, Southeast University, 210096 Nanjing, China. Email: bwxu@seu.edu.cn

individual within the population is evaluated by a pre-defined *fitness function* that gives higher values to better individuals. Then the population continuously evolves until the optimum is achieved, or another termination condition is satisfied. In the evolution process, the offspring population is generated from the parent population by several pre-defined *genetic operations*. Some common *code strategies* and *genetic operations* are shown in Table.1 [5].

Table.1: 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 a pair 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 the spin strategy

In ET, an individual in the population is one single test input, and the fitness function is constructed according to a particular test. The global optimal solutions are the test inputs that satisfy the test objective. In structural testing, the test goal is usually the execution of a statement (or a branch) of interest.

Since GA cannot promise to find the optimal solution, much attention has been paid on the improvement of the performance of ET [1, 10, 12, 13, 15, 16]. One of the core problems is how to configure the genetic operations used in ET. For different programs, the optimal configuration strategies are different.

However, even if there was a best static configuration strategy, GA could not promise to find the optimal solutions. One of the thorny problems is the population prematurity. A population is said to be *premature*, when the population has a uniform structure at all positions of the genes, without reaching an optimal structure in GA [5]. Thus, when the prematurity occurs, majority individuals in the population may have the same gene-type that corresponds to a local optimal solution. And, the evolution is sticking on the local optimal solution, and loses its power of searching the global optimal solutions.

Some researches [5] show that the prematurity also occurs in ET. If a test case has a quite high fitness value, it may occupy the whole population in only a few iterations rapidly. And the population may hold on this status without stepping towards the global optimal, until the permitted time is consumed.

In traditional ET, although proper configure strategies are helpful to prevent the population prematurity, there is a lack of self-adaptation methods in ET, which can recover the population diversity when the prematurity does happen.

3. Dynamic optimization strategy for evolutionary testing

From the above discussion, we know that the main reason of prematurity is the loss of diversity in a population. Some configuration strategies with lower selection pressure may prevent the occurrence of prematurity, since they have better effects on the diversity preservation. However these methods cannot recover the population diversity, when the prematurity happens.

It is feasible to introduce dynamic optimization strategies into ET. Some modules should be constructed to monitor the dynamic situation of the evolution process and take a series of actions to adjust the strategies, which can keep the evolution in a healthy status. These self-adaptation strategies may greatly improve the performance of ET. In this paper, we focus on the mutation operation to recover the diversity of a population.

Mutation is the occasional random alteration of a bit value of a gene, which alters some features with unpredictable consequences [5]. Mutation is an essential operation that maintains diversity in the population and prevents the population from prematurely converging on one local solution. The mutation possibility P_m is the degree that determines how often a bit value may be changed. If P_m is too low, it is difficult to maintain the diversity of a population, which may decrease the evolution efficiency. If P_m is too high, the heuristic search may degenerate to random search, which delays the location of the global optima.

The dynamic optimization strategy presented in the paper is denoted as **DOMP** (**D**ynamic **O**ptimization of **M**utation **P**ossibility). The basic idea of DOMP is that the mutation possibility should be adjusted dynamically according to the population. When phenomena of population prematurity are observed, P_m should be increased sharply to bring more various genes into the population. It helps to eliminate the mono-structural genes in the population, and break the stagnant status of the evolution. When the population diversity is recovered, P_m should be dropped to the ordinary value. Otherwise, the evolutionary search will degenerates to random search.

DOMP can be implemented as a monitor, which dynamically examines the status of the population. If the symptom of prematurity is detected, it will increase the mutation possibility to amend the evolution process. After the population is recovered, it drops the mutation possibility to the ordinary level.

DOMP has five parameters: observation frequency of DOMP *(OFD),* quantity of individuals in observations *(QIO),* proportion of repetitious individuals *(PRI),* persistency generation of adjustment *(PGA)*, and enhanced mutation possibility *(EMP)* in its configuration. The strategy observes the population every *OFD* generations. In once observation, it picks up *QIO* individuals. If the proportion of a certain gene-type exceeds the threshold value *PRI*, the strategy elevates the original P_m to *EMP*. And this status will be last for *PGA* generations of the evolution. It is critical to parameterize DOMP correctly, since DOMP brings extra cost into the evolutionary search. The parameters' value may have notable influences on the performance of ET.

OFD reflects the frequency that DOMP examines the population. It can be assigned with a value between 1 and the maximum of the iteration. The smaller its value is, the higher the frequency is. Apparently if *OFD* is very small such as 2 or 3, DOMP will examines the population in a high frequency, which may even decrease the performance of ET, since the population may not change greatly in such a few generations. In our experiments, we set *OFD* = 10, so that DOMP can detect the prematurity in time without much cost.

QIO defines the quantity of individuals that should be examined in once observation. It can be assigned with a value between 1 and the population size. The higher its value is, the more accurate the examination is. However, the examination also brings computing cost. Therefore, we set *QIO* =20 in our experiments to balance the cost and the accuracy.

PRI defines the threshold value of one certain gene-type's proportion in the population. The value is between 0.1 and 0.9. With a too low threshold, DOMP will be too sensitive to the prematurity, which will disturb the normal evolution process. However, a too high threshold will make DOMP too blunt to perceive the early symptom of the prematurity, until the evolution process is delayed badly. Therefore, the definition of the *PRI* is critical to DOMP.

PGA means how long the elevated P_m lasts for. It decides the intensity of DOMP to recover the population diversity. If its value is very low, DOMP may have no effects, since the inducement cannot last long enough to produce sufficient fresh genes. Conversely, if its value is too high, it will decrease the performance of ET, since evolutionary search becomes random search. Thus, the value of *PGA* is also important to DOMP.

EMP determines the value of the elevated P_m, which reflects the strength of the inducement. Its valid value is between 0.1 and 0.9. The higher its value is, the stronger the inducement is. A weak inducement cannot break the stagnant status of the evolution. However, a too strong inducement may change most genes in the same way, so that the original same genes would be changed into another gene-type together. It cannot recover the population diversity either.

4. Experiment study

The paper focuses on the structural evolutionary testing of a function without loop conditions. In our experiments, we utilize ET to generate test inputs that can cover a particular statement of a function.

4.1 Experiment 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*. We utilize the following indexes to evaluate the performances of different configuration strategies.

- **Hit Percent (P_h)**: the hit percent of ET. $P_h = T_h/T_0$, where T_h is the number of experiments that achieves the testing objective. It 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 generations of experiments that achieves the testing objective. This index reflects the convergence speed of the evolution. The lower HAG is, the faster ET is.
- **Total Test Cases (*TTC*)**: the average number of test cases generated in experiments.

$$TTC = Psize * [P_h * HAG + (1 - P_h) * IterMax]$$

The lower it is, the lighter ET is.

In our experiments, we utilize some benchmark parameters for ET, which are acquired from some previous experiments, and are not the key point of this paper. These parameters include: the mutation possibility P_m (assigned with 0.1), the survivability P_s (assigned with 0.5), T_0 (assigned with 150), *IterMax* (assigned with 250) and P_{size} (assigned with 100).

We tests three functions free of loops with different testing objectives (shown in Table.2) by structural ET.

- In **Triangle**, the condition is a *combination* of two Boolean expressions. Each expression is a simple equality whose requirement is easy to satisfy. However, the combination increases the complexity in a geometric progression.
- In **Quadratic**, the condition is an *algebraic equation*, which needs complex calculation.
- In **LineCover**, the condition is a combination of four Boolean expressions. Each expression is an equation whose requirement is harder to satisfy.

This testing objective is the most complex in our experiments.

In our experiments, the value range of each input variable in Triangle and Quadratic is [1, 200]. Since the testing objective of LineCover is very difficult to achieve, with the range of [1,200], ET will consume much more time to find the proper test cases in LineCover. Thus, the value range of each input variable in LineCover is [1, 50].

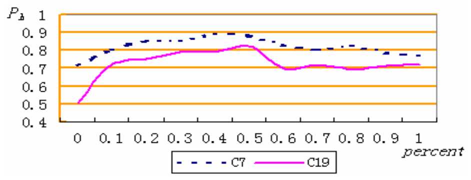
4.2 Parameter research of DOMP

We have conducted a series of experiments to investigate the optimal parameterization of DOMP by testing the three functions shown in Table.2. The experiment results show that the performance varies with the change of *PRI*, *PGA*, and *EMP*, which are critical parameters of DOMP.

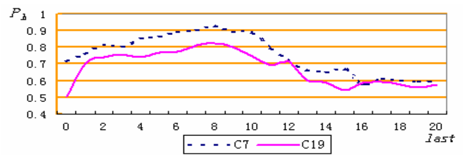
In LineCover, DOMP does not affect the performance of ET evidently, and the reason will be explained in the sections 4.3. However in Triangle and Quadratic, DOMP improves the performance of ET greatly. In this section, we choose two configuration strategies that have the most significant improvement to illustrate the parameterization of DOMP.

In Triangle, we choose configuration strategies C7 and C19, whose details are described in section 4.3. Figure.1 shows the variation of P_h with the change of *PRI, PGA*, and *EMP* in Triangle. From Figure.1, we can discover that P_h achieves the best value, when *PRI* is between 0.4 and 0.6, *PGA* is between 7 and 9, and *EMP* is between 0.6 and 0.8. Besides, our experiments with other configuration strategies provide consistent results with C7 and C19. Consequently, in Triangle, DOMP is parameterized with (*OFD*=10, *QIO* =20, *PRI* =0.5, *PGA* =8, *EMP* =0.7).

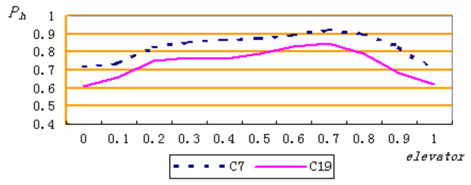
In Quadratic, we also choose configuration strategies C7 and C19. Figure.2 shows the variation of P_h with the changing of *PRI*, *PGA*, and *EMP* in Quadratic. It is discovered that, the same with Triangle, P_h achieves the peak value, when *PRI* is between 0.4 and 0.6 and *PGA* is between 7 and 9.



(a) Variation with *PRI*



(b) Variation with *PGA*



(c) Variation with *EMP*

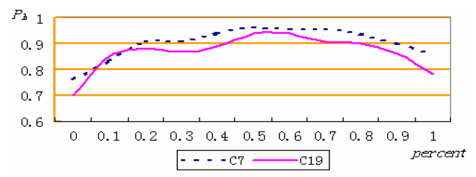
Figure.1: Ph variation with the parameters in Triangle

However, the best *EMP* is between (0.4, 0.6), which is lower than Triangle. And other strategies of Quadratic also present similar results with C7 and C19. Therefore, we can optimize the parameter by (*OFD* =10, *QIO* =20, *PRI* =0.5, *PGA* =8, *EMP* =0.5). It can be learnt in Figure.1 that the configuration is also fit for Triangle.

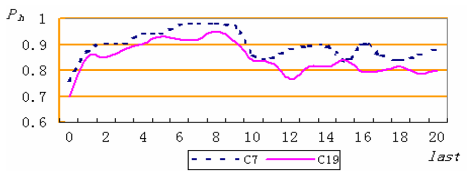
It is observed that the *EMP* of Triangle is higher than Quadratic. The reason may be that the condition to satisfy in Quadratic is a little more complex than the one in Triangle. Thus the evolution in Triangle is prone to have a higher convergence speed in the evolutionary search, which may be more likely to result in the population prematurity. Consequently, the *EMP* in Triangle should be stronger than the one in Quadratic, so that it could have more power to recover the population diversity.

Table.2: Different functions in experiments

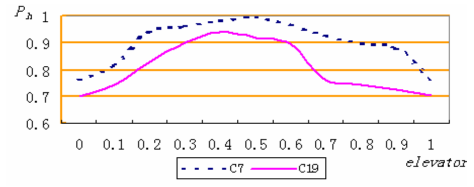
Function name	Input (integer)	Condition to satisfy (testing objective)	Corresponding result	Description of the function
Triangle	*a*, *b*, *c*	(*a* ==*b*) && (*b* ==*c*)	The triangle is an equilateral one.	*a, b* and *c* are the edges of a triangle. The program uses them to classify the triangle.
Quadratic	*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(*l1*x, *l2x*)) &&(*ry*+*h*==max (*l1y*, *l2y*))	The line segment is the diagonal of the rectangle.	*(l1x, l1y)* and *(l2x, l2y)* are two end points of a line segment. (rx, *ry)* is the bottom left point of a rectangle. *(w, h)* are the width and height of the rectangle. The program checks the position relationship between the line segment and the rectangle.



(a) Variation with *PRI*



(b) Variation with *PGA*



(c) Variation with *EMP*

Figure.2: P_h variation with the parameters in Quadratic

4.3 Performance improvement with DOMP

To examine the validity of DOMP, we compare the performances of ET with DOMP and without DOMP, by testing the three programs based on 24 configuration strategies that are listed in Table.3.

Table.3: Configuration strategies of ET

	Code		Selection		Crossover		Survival		
	B	G	R	S_1	S_2	U_1	S_3	F	U_2
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, S_1: Spin Selection, S_2: Single Crossover, U_1: Uniform Crossover, S_3: Spin Survival, F: Fitness Survival, U_2: Unrepeatable Survival.

Table.4 shows the performance comparison between ET with DOMP and ET without DOMP in Triangle. It is observed that P_h is increased obviously by DOMP in most cases. Figure.3 illustrates four configuration strategies that have greatest improvement in P_h, and the most significant one C19 has P_h elevated by 35.5%. Besides, it can be learnt in Table.4 that there is no configuration in which P_h is decreased by DOMP. However DOMP may increase *HAG* a little, when it gives significant elevation on P_h.

Table.4: Performance comparison between DOMP and Un-DOMP in Triangle

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

D: ET with DOMP; UD: ET without DOMP.

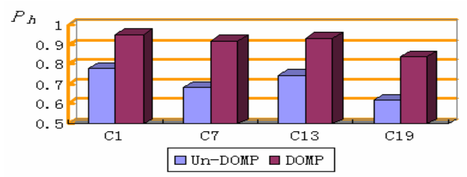


Figure.3: P_h comparison between DOMP and Un-DOMP in Triangle

Table.4 also supplies us with valuable data about the performances of different configuration strategies.

Figure.4 shows the comparison of P_h between Binary and Gray Code in three functions. And Figure.5 illustrates the comparison of P_h between Single and Uniform crossover in three functions. It is discovered that configurations with Gray Code and Single Crossover have better performances than the ones with Binary Code and Uniform Crossover.

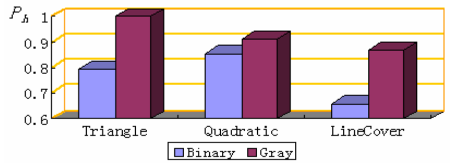


Figure.4: Comparison between Binary and Gray code

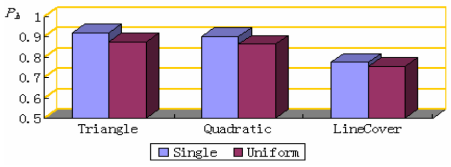


Figure.5: Comparison between Single and Uniform crossover

Besides, the best configuration such as C4, C5, C6, C10, C11 and C12 in Triangle, are composed of genetic operations with low selection pressure, such as Random Selection and Unrepeatable Survival. And the worst strategies like C13, C14, C15, C19, C20 and C21, always use methods with high selection pressure, such as Spin Selection and Spin Survival. This may due to the reason that the condition to satisfy in Triangle has a simple structure, which is easy to achieve by ET. The genetic operations with high selection pressure lead to a high convergence speed, which may easily cause the population prematurity in Triangle. Since the strategies with lower selection pressure are helpful for the preservation of the population diversity, it is advised to adopt them for testing objectives with simple structures and calculations.

In Quadratic, the performances of ET are also improved greatly by DOMP. And the most significant one is C19 in which P_h is elevated by 48.2%. The same with Triangle, the fittest configuration strategies for Quadratic are also the ones with low selection pressure, such as C4, C5, C6, C10, C11 and C12.

However, empirical data presented in Figure.6 show that DOMP has little influence upon LineCover. One possible reason is that the testing objective of LineCover is so complex that it is difficult for evolutionary search to find optimal solutions. Thus, the population may hardly converge prematurely and DOMP has no chance to optimize the evolution process. Our experiments also suggest that LineCover has different preference in configuration strategies. Figure.7 compares the performances of different genetic operations in Triangle, Quadratic, and LineCover.

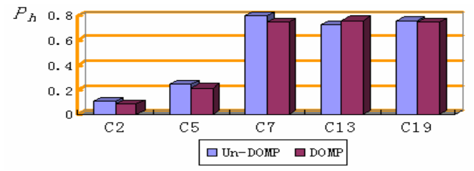
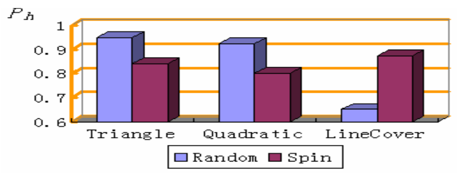
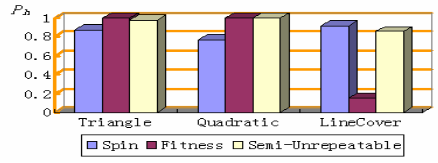


Figure.6: P_h comparison between DOMP and Un-DOMP in LineCover



(a) Comparison between Random and Spin Selection



(b) Comparison between Spin, Fitness, and Unrepeatable Survival

Figure.7: Performance comparison between different genetic operations

Figure.7 (a) shows that Spin Selection exceeds Random Selection dramatically in LineCover, while Random Selection has better performances in Triangle and Quadratic. Figure.7 (b) shows that Spin Survival who is worst in Triangle and Quadratic exceeds other survival methods in LineCover. These results advise that genetic operations with high selection pressure are more suitable for LineCover. We think it may due to the reason that the condition to satisfy in LineCover is much more complex than the ones in Triangle and Quadratic. Unfortunately, the diversity conservation strategies may decrease the speed further, which make it is even harder to hit the target statements. In this condition, the genetic operations with high selection pressure may accelerate the search to improve the performances of ET.

In conclusion, DOMP loses its power in LineCover, since there is rarely prematurity in the evolution process. We can improve the performance of this kind of object conditions by increasing the selection pressure appropriately, which may help to speed up the convergence. It is also helpful to increase the iteration number of the evolution, however it may consume much more testing resources.

4.4 Empirical conclusions and advices

In this section, we present some advices for structural ET, which are obtained from the empirical data in section 4.2 and section 4.3. It is notable that the advices are only for programs with no loops, and for ET whose testing objective is a single statement.

1. It is advisable to utilize DOMP in practical testing. Our empirical results show that DOMP can improve the performance of ET significantly in many cases and do no harm to the evolution process. Although it does not promise to improve the performance in all cases, it is practical to turn on DOMP to grasp the chance of optimization, especially when it is the first time to test a new function.

2. DOMP's configuration may be adjusted according to object conditions under test. In our experiments, the optimal parameterization of DOMP for Triangle is (*OFD* =10, *QIO* =20, *PRI* =0.5, *PGA*=8, *EMP* =0.7), and for Quadratic is (*OFD* =10, *QIO* =20, *PRI* =0.5, *PGA*=8, *EMP* =0.5). By studying the empirical data carefully, we suggest that it is feasible to initialize DOMP with (*OFD* =10, *QIO* =20, *PRI* =0.5, *PGA*=8, *EMP* =0.5).

3. For programs with simple condition to satisfy, such as simple algebraic equations and combinations of several simple Boolean expressions, it is strongly advised to use DOMP in ET, since DOMP may improve P_h significantly for these programs.

It is wise to be careful about combinations of Spin Select, Spin Survival, and Uniform Crossover in these programs, which provide high selection pressure. Such kinds of combinations may lead to poor performances of ET.

4. In programs with complex conditions to satisfy, DOMP may not be as effective as in the programs with simple conditions. However, DOMP is also suggested to use, since it provides more protection for ET from losing the population diversity.

For this kind of programs, genetic operations with high selection pressure, such as Spin Select and Spin Survival, are preferred. And we should be careful of combinations of Uniform Crossover, Random Selection, and Fitness Survival, which have low selection pressure.

In these programs, increasing the selection pressure and the number of iteration may help to elevate P_h, however the increased number of iteration can also consume much more testing resources.

5. It is advised to prefer Gray Code to Binary Code, and prefer Single Crossover to Uniform Crossover. The experiment data show that the former ones may provide better performances of ET in most cases.

5. Related works

Other researches have investigated ET in structural testing. H. Sthamer et al. propose a technique that generates test cases with GA for structural testing using Control Flow Graph [5]. And R. Pargas et al. utilize the Control Dependence Graph to evaluate a test case in the population, which makes ET a full-automated testing technique [16].

Since ET has been widely researched in different applications [8], the performance of ET becomes more critical. Some researchers have proposed different methods for the performance optimization of ET. For example J. Wegener et al. combine the approximate level with the condition to construct a more efficient fitness function [7]. M. Harman et al. use the amorphous slicing to remove the flag variables, which facilitate the construction of better fitness functions [12, 15].

Our strategy DOMP focuses on the dynamic part of the performance optimization of ET, which is complementary for the existing methods. It can evidently improve the performance of ET in many cases. Besides, the basic advices we proposed are helpful to facilitate the configuration of ET.

6. Conclusions

In this paper, we present the idea of dynamic optimization DOMP to improve the performance of ET. It contributes to recover the diversity of population from the prematurity. And the empirical results show that the strategy can greatly improve the performance of ET in many cases.

Besides, we also provide some advices obtained from experiments for the configuration of ET, which are helpful to configure ET in practice.

In future works, we will continue the research on the dynamic optimization for ET. We plan to extend the applications of the dynamic optimization strategies, which could solve more problems such as the input domain restriction. And we will carry out more experiments to test different programs, such as functions with loops, to verify the conclusions in this paper, and acquire more experience on performance optimization of ET. With sufficient empirical data, we will try to propose some systematic rules in this field.

7. References

[1] 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.

[2] B. Jones, H. Sthamer, D. Eyres. "Automatic structural testing using genetic algorithm", Software Engineering Journal, vol.11.no.5.pp299-306, 1996.

[3] C. Ghezzi, M. Jazayeri, and D. Mandrioli. "Fundamentals of Software Engineering", Prentice Hall, Englewood Cliffs, NJ, USA, 1991.

[4] C. Michael, G. McGraw, M. Schatz, and C. Walton. "Genetic Algorithms for Dynamic Test Data Generation. Technical

Report", RSTR-003-97-11. Reliable Software Technologies Corporation (1997).

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

[6] H. Sthamer, J. Wegener and A. Baresel, "Using Evolutionary Testing to improve Efficiency and Quality in Software Testing", Proceedings of the 2nd Asia-Pacific Conference on Software Testing Analysis & Review, Melbourne, Australia, 2002.

[7]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.

[8] J. Wegener, H. Sthamer and A. Baresel "Application Fields for Evolutionary Testing", Eurostar 2001 Stockholm, Sweden, November 2001.

[9] 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.

[10] J. Wegener and O. Bühler, "Evaluation of Different Fitness Functions for the Evolutionary Testing of an Automatic Parking System", in The Genetic and Evolutionary Computation Conference, Seattle, Washington, pp: 1400-1412, 2002.

[11] 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, 18th 2002.

[12] M. Harman, L. Hu, R. Hierons, A. Baresel, and H. Sthamer, "Improving Evolutionary Testing by Flag Removal", in The Genetic and Evolutionary Computation Conference, New York, USA, pp. 1359-1366, July 2002.

[13] M. Harman, L. Hu, R. Hierons, J. Wegener, H. Sthamer, A. Baresel and M. Roper, "Testability Transformation", IEEE Transactions on Software Engineering", Vol.30, No.1, January 2004.

[14] N. Tracey, J. Clark, J. McDermid, and K. Mander: Integrating Safety Analysis with Automatic Test-Data Generation for Software Safety Verification. Proceedings of the 17th International System Safety Conference, pp. 128-137 ,1999.

[15] P. McMinn and M. Holcombe, "The state problem for evolutionary testing", in The Genetic and Evolutionary Computation Conference, Chicago, USA, 2003.

[16] R. Pargas, M. Harrold and R. Peck, "Test-Data Generation Using Genetic Algorithm. Software Testing", Verification & Reliability, 9 (4): pp.263-282, 1999.