Application of Improved Harmony Search Algorithm in Test Case Selection

Ming Huang
Software Technology Institute, Dalian Jiao Tong University, Dalian, China
Email: dlhm@263.net

Shujie Guo
Mechanical Engineering Institute, Dalian Jiao Tong University, Dalian, China

Xu Liang
Software Technology Institute, Dalian Jiao Tong University, Dalian, China

Xuan Jiao
Management Institute, Dalian University of Technology, Dalian, China

***Abstract*—Regression test is an effective means to ensure the quality of software. But the test efficiency will become lower and lower as the test case library becomes increasingly large in the test. Therefore, in order to improve the efficiency and quality of regression test, it is necessary to implement the test case safety reduction. The optimization selection of the test case is an effective means of the test case safety reduction. In order to solve the problem of test cases optimization selection, this paper proposes a test case selection method based on the improved harmony search algorithm. Specifically, the researcher adds an excellent harmony element library based on the classical harmony search algorithm for storing the searched excellent test case; meanwhile, he also improves the way of generating the new harmony to allow these excellent test cases to be retained with a certain probability. In the search process, the researcher makes the adaptive adjustment of the algorithm parameter BW in a non-linear incremental manner, improving the global search capability of the algorithm. In addition, the excellence protection strategies are also increased to prevent against the degradation in the optimization search process. In order to verify the feasibility of the improved algorithm, the researcher writes four programs: the genetic algorithm, differential evolution algorithm, classical harmony algorithm and improved algorithm, and conducted the comparative experiment. The experimental result shows that the improved algorithm significantly improves the optimization search performance compared with the classical harmony search algorithm and has the better performance compared with the genetic algorithm and differential evolution algorithm.**

***Index Terms*—regression test; test case selection problem; harmony search algorithm.**

I. Introduction

The regression test refers to the re-test on the existing software which has been modified to confirm that there is no a new error appearing or there is no side effect on the existing unmodified module when making amendment [1]. The regression test, as an integral part of software life cycle, accounts for a large proportion in the whole software test process, which has important significance in assuring the software quality. In the software life cycle, in order to fix software bugs or respond to customers' demands for change, the software will be frequently revised and new software versions continue to be introduced, resulting in the dramatic increase of the accumulated test cases in the test case library. As the changed module is limited in a modification, there will be a considerable number of test cases in the current baseline test case library which are insensitive to the modified module and have no ability to explore errors. In order to improve the efficiency of regression test, for this revision, we must select the test case suite from the test case library which has stronger ability of exploring regression errors; this is the test case selection problem. Currently, the test case selection techniques the researchers have proposed generally fall into the following categories: Integer Programming, Heuristic Greedy Search, Intelligent Optimization Search Algorithm, Graph Walk based Approach, Firewalls Method [2-7]. These test case selection techniques have some deficiencies in application scope, computational efficiency, the algorithm performance and other aspects. To address these shortcomings, this paper proposes an improved harmony search algorithm — Harmony Search Algorithm with Excellent Element Library (EELHS), to complete the selection of the test cases in the regression test.

II. Test Cases Selection

A. Definitions

Definition 1. Sensitive function: A program covers a large number of functions, which will be given different attentions at a regression test; the function which is given the particular attention at a test is called "sensitive

Project supported by the National High Technology Research and Development Program of China (No. 2012AA041402-4);
Manuscript received July 23, 2013; revised September 11, 2013; accepted September 14, 2013.

doi:10.4304/jsw.9.5.1170-1176

function". The common sensitive functions include the following categories: newly modified function, coupling function of the newly modified function, function of the selling point of the program, function of the most fatal part of the program and the more vulnerable function of the program.

Definition 2. The function test coverage: The test coverage of the function F: TCoverageF=|TCF|/|TTotal|, where |TCF| is number of the test case of covering function "F" and |TTotal| is the number of test cases.

Definition 3. Test case selection problem: Suppose: the old version of the program is P, the new version of the program is P', which has "m" sensitivity functions MF={f_1, f_2, ... f_m}; the original baseline test case library T0 has n test use cases T0={t_{01}, t_{02}, ... t_{0n}}. The test case selection problem refers that we select r test cases from the test case library T0 to form a test suite ST_r = {t_{s1}, t_{s2}, ... t_{sr}} in order to test P' so that the test coverage of the MF can be as good as possible and the test cost can be as low as possible.

B. Test Suite Evaluation Criteria

According to the requirements of the test case selection problem, we formulate the criteria to evaluate on the advantages and disadvantages of the test suite: sensitivity function test coverage criteria and test cost criteria.

Sensitivity function test coverage criteria: The bigger the average test coverage of sensitivity function is, the better it is; the smaller the mean square error of the test coverage of the sensitive function is, the better it is; namely, various sensitive functions should have the relatively high and uniform test coverage.

Test cost standard: The smaller the sum of execution time of test cases in the test suite is, the better it is.

C. The Mathematical Model of the Test Case Selection Problem

Based on the above analysis, the test case selection problem can be expressed as a multi-objective programming problem.

Suppose: the set of sensitive functions in this test is $F_{Sensitive}$={f_{S1},f_{S2},…., f_{Sp}}, the test coverage of the function f_{Si} is TC_{fSi}; the original baseline test case library T0 has *n* test cases T0={t_{01},t_{02},…t_{0n}};the execution time of the test case T_i is T_{ti}; the selected test suite ST_r={ t_{s1},t_{s2},…t_{sr} }. In this case, the mathematical model of the test case selection problem is as follows:

$$\max \quad \overline{TC}(x) = \frac{\sum_{i=1}^{p} TC_{fSi}(x)}{p}$$

$$\min \quad \delta(x) = \sqrt{\frac{1}{p}\sum_{i=1}^{p}(TC_{fSi}(x) - \overline{TC}(x))^2}$$

$$\min \quad T_{\text{summation}}(x) = \sum_{i=1}^{r} T_{t_{si}}(x)$$

In this formula, p is the size of the set of sensitive function $F_{Sensitive}$; $\overline{TC}(x)$ is the average value of the test coverage of the sensitive functions; $\delta(x)$ is the mean square error of test coverage of the sensitive functions; $T_{\text{summation}}(x)$ is the total execution time for the test cases. On the basis of standardizing the targets, the problem can be converted to the following model through the linear weighing:

$$\max F(x) = \lambda_1 \overline{TC}(x) - \lambda_3 \delta(x) - \lambda_3 T_{summation}(x)$$

$$\lambda_i \in [0,1](i = 1,2,3) \qquad \sum_{i=1}^{3} \lambda_i = 1$$

III. HARMONY SEARCH ALGORITHM

The harmony search algorithm is an emerging heuristic search algorithm, which was first proposed in 2001 by Zong Woo Geem after he was inspired by the improvisation of musicians [8]. As the harmony search algorithm uses the random search rather than the gradient search, it solves the problem without any gradient information, and it has the advantages of strong versatility and easy implementation. The harmony search algorithm has been successfully applied to solve different scientific and engineering problems [9], such as structural design [10-11], the pipe network design [12], scheduling problems [13], groundwater management [14], the path planning [15] and reliability optimization problem [16] and so on.

The optimization process of the harmony search algorithm includes five steps: initializing the algorithm parameters, initializing the harmony memory, improvising a new harmony, updating the harmony memory and test stopping criteria.

Step 1: Initialize the algorithm parameters.

The main task is to complete the determination of the objective function and configure the algorithm parameters. The algorithm parameters include harmony memory size (HMS, that is, the number of solutions each generation retains, similar to the population size in the genetic algorithm), harmony memory considering rate (HMCR) , pitch adjusting rate (PAR), bandwidth (BW), and the number of improvisations (NI).

Step 2: Initialize the harmony memory

In step 2, use the randomly generated HMS solution vector to initialize the harmony memory HM.

Step 3: Improvise a new harmony

Each component of the new harmony $X' = (x_1', x_2', ..., x_N')$ is generated in three ways: select one randomly from the range of decision variables with the probability (1-HMCR); select from the memory with the probability HMCR; select from the memory with probability HMCR×PAR and make the pitch adjustment.

$$x_i^{new} \leftarrow \begin{cases} x_i(k) \in \{x_i(1), x_i(2), ..x_i(k_i)\} & w.p. \quad R_{Random} \\ x_i(k) \in \{x_i^1, x_i^2, ..x_i^{HMS}\} & w.p. \quad R_{Memory} \\ x_i(k \pm BW) & w.p. \quad R_{Pitch} \end{cases}$$

In this formula, R_{Random} = 1-HMCR, R_{Memory} = HMCR, R_{Pitch} = HMCR×PAR.

Step 4: Update the harmony memory.

Make evaluation on the new harmony "X_{new}" obtained at Step 3: if X_{new} is superior to the worst harmony X_{worst} in the current harmony memory and the harmony the same as X_{new} doesn't exist in the harmony memory, replace X_{worst} with X_{new}.

Step 5: Test stopping criteria

Judge whether to meet the stopping criteria; if so, report the optimization results; otherwise, implement Step 3 and 4 in a circular way.

IV. EELHS-BASED TEST CASE SELECTION

The harmony search algorithm mainly has the following weaknesses: In the first place, the number of the algorithm parameters is relatively large, the relationships between parameters are more complex, and the search performance has stronger sensitivity and dependence on the parameters. Secondly, it is difficult to obtain the global optimal solution within a limited period of time, so the global search performance needs to be enhanced. In response to these weaknesses, the researchers have proposed many improved algorithms, which mainly concentrate in three aspects: improving the algorithm parameter configuration mode [11, 17-21], combining with other algorithms to form the hybrid harmony search algorithm [16] and add new operators [22]. To further improve the efficiency of the harmony search algorithm in solving the test case selection problem, the harmony search algorithm is improved in three aspects based on the research result of other researchers in combination with the characteristics of the test case selection problem, forming the EELHS.

A. Harmony Search Algorithm with Excellent Element Library(EELHS)

Definition 4. Harmony element: In the harmony search algorithm, a solution vector of the problem to be solved constitutes a harmony; a component $x_i(i=1..N)$ in the harmony $X=(x_1,x_2,...,x_N)$ is called "harmony element".

The classical harmony search algorithm is improved through EELHS from three aspects.

Improvement 1: Add the excellent harmony element library, and make a corresponding improvement of the generation way of the new harmony in order to preserve the relatively excellent harmony, improving search performance of the algorithm.

Introduce an Excellent Element Library (EEL) storing the excellent harmony elements for the algorithm. The excellent harmony element consists of two parts: the intersection of the set composed of the elements of the harmony with the optimal total target and the set composed of the suboptimum harmony elements; the intersection of the set composed of the elements of the harmony with the optimal sub-target and the set composed of the suboptimum harmony elements. We update the optimal harmony library once every certain search generation. Let's take the test case selection problem as an example. In case of the search at the i-th generation, the harmony with the best total target value is TBs={tbs1, tbs2,…., tbsn}, the harmony with the second best total target value is TBt={tbt1, tbt2,…., tbtn}; the harmony with the best average test coverage $\overline{TC}$ is TCBs={tcbs1, tcbs2,…., tcbsn}, the harmony with the second best average test coverage variance $\overline{TC}$ is TCBt={tcbt1, tcbt1,…., tcbtn}; the harmony with the best mean square error of the coverage (δ) is ΔBs={δbs1, δbs2,…., δbsn}, the harmony with the second best mean square error of the coverage (δ) isΔBt={δbt1, δbt2,…., δbtn}; the harmony with the best total execution time is TiBs={Timebs1, Timebs2,…., Timebsn}, the harmony with the second best total execution time is TtBt={Timebt1, Timebt2,…., Timebtn}. The formula of EEL is as follows:

$$EHL=((TBs\cap TBt)\cup(TCBs\cap TCBt)\cup(\Delta Bs\cap\Delta Bt)\cup(TiBs\cap TiBt))$$

On the basis of introducing the EEL, the generation of the new harmony is completed in the following ways.

When the EEL is empty, the new harmony is generated in the same way as the classic harmony; when the EEL is not empty, the generation method of each component of the new harmony increases from three kinds to four kinds: select from the memory with the probability HMCROriginally; select from EEL with probability HMCRBest; randomly select one from the range of decision variable values with the probability 1-HMCRBest-HMCROriginally; select from the memory or EEL with the probability HMCRBest+ HMCROriginally × PAR and make the pitch adjustment.

$$x_i^{new}\leftarrow\begin{cases}x_i(k)\in\{x_i(1),x_i(2),..x_i(k_i)\} & w.p.\quad R_{Random}\\ x_i(k)\in\{x_i^1,x_i^2,..x_i^{HMS}\} & w.p.\quad R_{Memory}\\ x_i(k)\in\{x_i^1,x_i^2,..x_i^{EELS}\} & w.p.\quad R_{\text{Excellent}}\\ x_i(k\pm\text{BW}) & w.p.\quad R_{Pitch}\end{cases}$$

In this formula, R_{Random} = 1-HMCRBest-HMCROriginally, R_{Memory} = HMCROriginally, $R_{\text{Excellent}}$ = HMCRBest , R_{Pitch} = (HMCRBest+HMCROriginally)×PAR; EELS represent the number of harmony in the excellent harmony library.

Improvement 2: Automatically adjust the algorithm parameters with the search generations.

The harmony search algorithm is more sensitive to the algorithm parameters. In order to overcome the lack of fixed parameters and improve the search performance of the algorithm, we add the adaptive adjustment functions of parameters to the improved algorithm. The improved algorithm includes three core parameters, namely, HMCRBest, PAR and BW (i.e. the step size at the time of pitch adjustment). In order to determine the adaptive adjustment functions of the parameters of the improved algorithm, we experiment many times with each core parameter adopting some adjustment functions such as

linear increment, linear decrement, non-linear increment and non-linear decrement; meanwhile we also make the portfolio adjustment experiment with the parameters PAR and BW. In the experiment, each configuration independently runs 50 times. According to the mean and standard deviation of the obtained optimal objective function value, we judge the performance of configuration solution. The experimental results show that, for the regression test cases selection problem, whether the incremental adjustment method or the decremental adjustment method is adopted, the adaptive adjustment of the parameters "HMCRBest" and "PAR" cannot improve the search performance of the algorithm. The adjustment of the parameter "BW" in the way of decrement cannot improve the performance of the algorithm; in the incremental way, BM can improve the algorithm performance, and the non-linear incremental adjustment is better than the linear incremental adjustment. In view of this, in the improved algorithm, the following functions are used to make the adaptive adjustment of BW; other parameters don't make adjustment.

$$BW(gn)=\frac{MaxBW-1}{MaxGn^3-1}gn^3+\frac{MaxGn^3-MaxBW}{MaxGn^3-1}$$

In this formula, MaxBW stands for the supposed maximum value of the BW parameter; MaxGn stands for the search stopping generation.

Improvement 3: Increase the excellence protection measures to prevent against the degradation in the optimization search process.

Without the measures for excellence protection, the classical harmony search algorithm degradation may occur, resulting in the loss of excellent individuals obtained in the previous search. To avoid this phenomenon, the excellence protection strategy is added to the improved algorithm.

The process flow of the improved algorithm is shown in Figure 1.

B. EELHS-based Test Case Selection Problem Solution

The EELHS-based test case selection problem solution includes four major steps: 1. test case entry; 2. analysis of code change; 3. test case optimization selection; 4. test coverage analysis.

Step 1: Test case entry

Before testing, first, use the source code instrumentation to treat the test program. The probe inserted in the source code is used to collect the information of test cases and store the captured test case execution path, execution time and other relevant information in the database. These practical operational information creates the conditions for the optimization selection of test cases and their coverage analysis. Based on the source code instrumentation, the test case entry work can be completed through running the test cases in the original baseline test case library T0.

Step 2: Code change analysis

The code change analysis refers that the program change report and sensitive function information are obtained through the comparison of old and new versions in the program code. Generally speaking, the code change analysis is divided into four kinds: namely, packet level, class level, function level and statement level. For the package level and class level, as the granularity compared is too coarse, it will affect the quality of regression test optimization. For the statement level, as the granularity compared is the smallest, the analysis result obtained is the most accurate, and the regression test has the highest quality. However, compared with the function-level change analysis, the quality difference of regression test is very limited; in addition, as it causes the excessive cost in the execution time, the efficiency of regression analysis will be affected. Therefore, we use the function-level change analysis as the change granularity of the regression test.

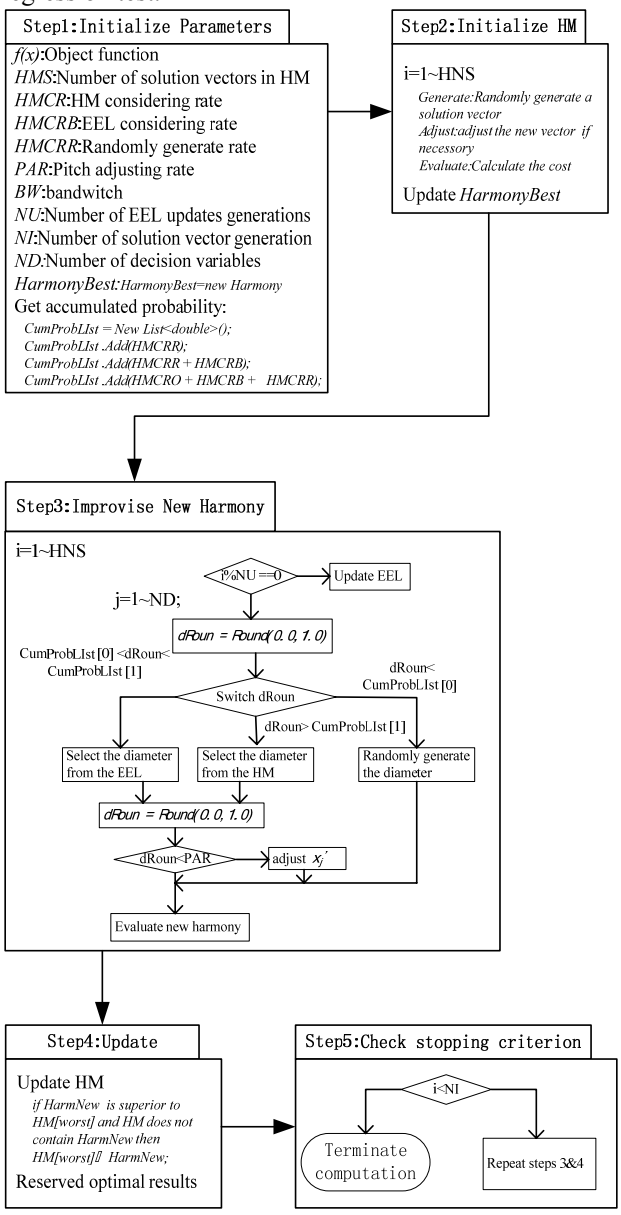

Figure 1. The Processing Flow of EELHS

In the code change analysis, the function is taken as the unit. First, decompose the code into individual functions, and normalize the statements in the function. The main

purpose of statement standardization is to format the valid statements, that is, form the format of a statement in each row, so that the differences can be compared. The main work of statement standardization mainly includes five aspects: first, removing the comment statement; second, statement decomposition, that is, decompose multiple statements existing in one line into the format of one statement in one line; third, statement composition, that is, merge the statements spreading over multiple lines into one line; fourth, treat the braces, that is, isolate the braces "{"and "}"in the statement and make them form a line independently; fifth, judgment statement processing, that is, if the keyword "else" follows the keyword "if" in one line, the "if" statement shall be separated out to form an independent line. Secondly, obtain the changed function in the code, i.e. *FunChanged*, through the static analysis of source files of old and new versions after normalization treatment; then, search the function calling "FunChanged" in the code and get the function "*FunCoupling*" having the coupling relationship with the change function; finally, through the communication with the developers, obtain the functions of part of the selling points of the program, the function of the fatal part of the program, the function of the more vulnerable part of the program and other sensitive functions.

Step 3: Test case optimization selection

In accordance with the mathematical model of the test case selection problem given in Section 2.3, we use EELHS to complete the search work of excellent test cases. It shall be noted that, in the new harmonies generated in STEP3, the same harmony element may appear more than once in a certain harmony, that is, the same test case is selected repeatedly; in this case, such harmony cannot meet the requirements, so the adjustment is necessary. The adjustment method is as follows: first, obtain the sequence of sensitive functions, i.e. Funs = (F1, F2, ..., Fn), in accordance with the ascending order of test coverage; then, according to the sequence of from F1 to Fn, sequentially select the test cases covering the function in "Funs" to replace the same harmony element until this harmony meets the requirements. This adjustment method can guarantee that the functions with the lower test coverage can be tested comprehensively.

Step 4: Test coverage analysis

According to the execution paths of the selected test cases, submit the coverage analysis report so that the testers can compile or modify the test cases based on it.

V. Experiments and Analysis

A. Value of HMCRBest

According to the research result of ZONG WOO GEEM, the ideal value range of the parameter HMCR is 0.7 - 0.95 [23]. In order to determine the optimal value solution of HMCRBest, set the value of HMCRBest respectively as 0.05, 0.02, ..., 0.30, and then calculate 10 times for each value and record the average value of the objective functions found in the 10 optimization processes. The experimental result is shown in Figure 2.

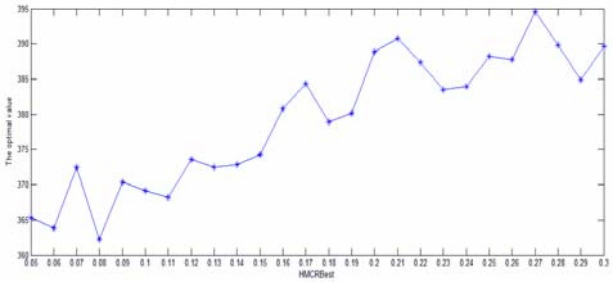

Figure 2. The Influence of the HMCRBest Value on the Optimization Result

As is shown in Figure 2, when the HMCRBest value is less than or equal to 0.2, the fitness of optimal solution increases with the increase of HMCRBest; when the HMCRBest value is greater than 0.2, the fitness of the optimal solution tends to stabilize; when the HMCRBest value is 0.27, the fitness of the best individual obtained is the largest. Thus, the ideal value range of HMCRBest is 0.20-0.30.

B. Comparison with Other Algorithms

In order to test the validity of the EELHS algorithm in solving the test case selection problem, we wrote four sets of software: standard genetic algorithm (GA), standard differential evolution algorithm (DE), standard harmony search algorithm (SH) and harmony search algorithm with excellent element library (EELHS), and made the comparative experiment in the same software and hardware environment. In the experiment, each algorithm respectively ran 100 times. After that, we made the comparison in the mean and mean square error of the maximum objective function values found by algorithms, the maximum objective function values found by algorithms and the optimization process to find the best value.

Scheme Design:

Experimental data: The code to be tested contains a total of 200 functions, of which 49 sensitive functions are our concern; in the original baseline test case library Ti, there are 5000 test cases, of which 4,606 test cases cover the sensitive functions of this test. It is required that 200 test cases shall be selected from 4606 test cases for this regression test.

Experimental platforms: Intel Core i3 2.13GHZ CPU, 2.0GB RAM, Windows 7 operating system.

Test software development platform: Microsoft Visual Studio 2010.

Experimental Results:

The maximum objective function values found by algorithms in 100 operation processes and their means and mean square errors are shown in Table 1.

TABLE I.

ALGORITHM PERFORMANCE COMPARISON TABLE

algorithm	Mean of the best fitness	MSE of the best fitness	Best fitness
GA	348.1183	9.1324	366.860490774905
DE	352.2584	7.3098	378.077184939278
HS	350.4941	8.3399	366.456281001912
EELHS	384.2363	9.9892	416.290332829723

The distribution of the maximum objective function values found by algorithms in 100 operation processes is shown in Figure 3.

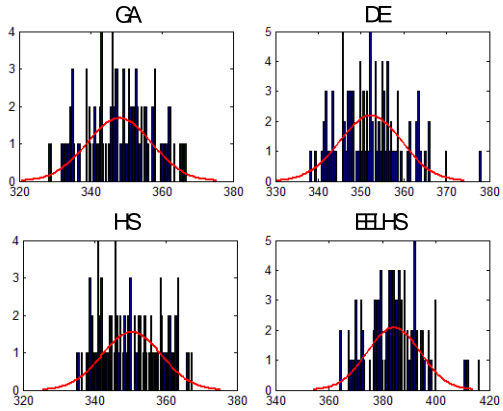

Figure 3. Distribution Comparison of the best fitness

It can be seen from TABLE Ⅰ and Figure 3 that the EELHS can find greater best fitness with satisfactory mean square errors.

The optimal evolution processes of algorithms in 100 operation processes are shown in Figure 4.

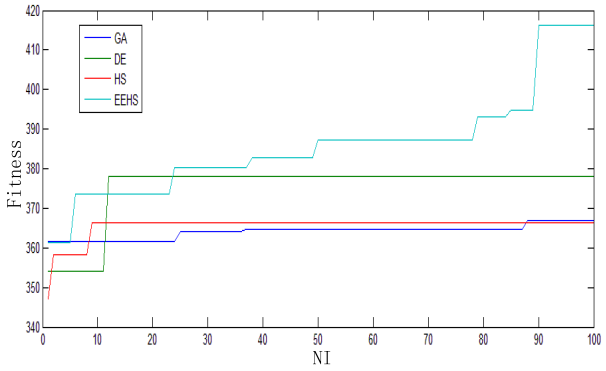

Figure 4. Comparison of Optimal Evolution Processes of Algorithms

Result Analysis:

Table 1 shows that the average value of the maximum objective functions searched by EELHS is significantly better compared to other three algorithms. The maximum objective function values searched by EELHS mainly concentrate between 370 and 390, and those searched by other three algorithms concentrate between 340 and 360. The EELHS algorithm is able to search the better solutions for two reasons: first, the way of generating new harmony of the algorithm can make the excellent harmony elements searched by many generations be retained so that the newly generated harmony can be relatively good; second, as the algorithm has stronger continuous optimization ability, it is not easy to fall into the state of local optimum, which can be seen clearly from the figure of Comparison of Optimal Evolution Processes of Algorithms. Throughout the search process, the EELHS algorithm has always maintained an evolutionary trend towards the global optimal solution while other algorithms fall into the "premature" state after the 10th generation. The continuous optimization capability of the algorithm is mainly due to the nonlinear incremental dynamic adjustment strategy of the algorithm BW. At the middle and later stage of optimizing the search process, the BW parameter will become relatively larger, increasing the toning step size in the process of generating new harmony. Thus, the algorithm can have the broader scope of search, improving the global search capability of the algorithm.

VI. Conclusion

The regression test, as a part of the software life cycle, accounts for a large proportion of the workload in the whole software test process as it is conducted repeatedly at all stages of the software development. In the software life cycle, even if a test case library is well-maintained, it may become quite large, which makes it impractical that all the test cases re-run for each regression test. In order to reduce the cost of regression test and improve test efficiency, it is necessary to conduct the safety reduction for the test cases, specifically, select the test cases with the close relationship with the concerned codes to complete the test work. This paper proposes a test case selection problem solution, which uses the improved harmony search algorithm to optimize the selection of test cases. Compared with the genetic algorithm, differential evolution algorithm and classical harmony search algorithm, this algorithm has better search performance. The next step of work is to study the automatic generation of test cases covering the specified functions, namely, according to the function coverage analysis report, for the function with the lower test coverage, i.e. Fi, automatically generate the test case suite SetTSFi, making the test cases in SetTSFi cover the function Fi.

Acknowledgment

Thanks National High-Tech Research and Development Program of China (863 Program) for its funding of sub-project "Testing of the operating system kernel module and software library" (No. 2012AA041402-4). We are grateful to other members of the project team for their valuable help and suggestions.

References

[1] S. Raju, G. V. Uma Factors Oriented, "Test Case Prioritization Technique in Regression Testing using Genetic Algorithm", *European Journal of Scientific Research*,vol.74 no.3, pp.389-402, 2012.

[2] GUQing, TANGBao, CHENDao-Xu, "A Test Suite Reduction Technique for Partial Coverage of Test Requirements", *Chinese Journal of Computers*, vol.34 no.5: pp.879-887. 2011

[3] Bharti Suri, Isha Mangal, "Analyzing Test Case Selection using Proposed Hybrid Technique based on BCO and Genetic Algorithm and a Comparison with ACO", *International Journal of Advanced Research in Computer Science and Software Engineering,* vol.2 no.4: pp.206-211, 2012.

[4] Zheng, J., Williams, L., Robinson, B. , et al.. "Regression test selection for black-box dynamic link library components", *Proceedings of the Second International Workshop on Incorporating COTS Software into Software*

Systems: Tools and Techniques, Minneapolis, MN, USA: pp.9–14, 2007.
[5] Ying-Dar Lin, Chi-Heng Chou, Yuan-Cheng Lai, et al.. "Test coverage optimization for large code problems", *Journal of Systems and Software*, vol.85 no.1: pp.16–27, 2012.
[6] S. Yoo, M. Harman, "Regression testing minimization, selection and prioritization: a survey", *Software Testing, Verification and Reliability*, vol.22 no.2: pp.67–120, 2012.
[7] Monthawan Raengkla, Taratip Suwannasart. "A Test Case Selection from Using Use Case Description Changes", *Proceedings of the International MultiConference of Engineers and Computer Scientists 2013, Hong Kong:* pp.507-510, 2013.
[8] Z.W. Geem, J.H. Kim, G.V. Loganathan, "A new heuristic optimization algorithm: harmony search", *Simulation,* vol.76 no.2: pp.60–68, 2001.
[9] Z.W. Geem, *Recent Advances in Harmony Search Algorithm.* Berlin, Springer, 2009.
[10] M.P. Saka. "Optimum design of steel sway frames to BS5950 using harmony search algorithm", *Journal of Constructional Steel Research,* vol.65 no.1: pp.36–43, 2009.
[11] S.O. Degertekin, " Improved harmony search algorithms for sizing optimization of truss structures", *Computers and Structures,*vol.92 no.93: pp.229–241, 2012.
[12] Z.W. Geem, " Particle-swarm harmony search for water network design" *Engineering Optimization*, vol.41 no.4: pp.297–311, 2009.
[13] Z.W. Geem, " Multiobjective optimization of time-cost trade-off using harmony search", *ASCE Journal of Construction Engineering and Management*, vol.136 no.6: pp.711–716, 2010.
[14] M. Tamer Ayvaz, "Application of harmony search algorithm to the solution of groundwater management models", *Advances in Water Resources* vol.32 no.6 pp.916–924, 2009.
[15] Z.W. Geem, K.S. Lee, Y. Park, "Application of harmony search to vehicle routing", *American Journal of Applied Sciences* vol.2 no.12 pp.1552–1557, 2005
[16] Dexuan Zou, Liqun Gao, Jianhua Wu et al.. "An effective global harmony search algorithm for reliability problems", *Expert Systems with Applications*, vol.38: pp.4642–4648, 2011.
[17] Majid Jaberipour , Esmaile Khorram, "Two improved harmony search algorithms for solving engineering optimization problems", *Commun Nonlinear Sci Numer Simulat*, vol.15: pp.3316–3331, 2010.
[18] Dexuan Zou, Liqun Gao, Steven Li,Jianhua Wu, "Solving 0–1 knapsack problem by a novel global harmony search algorithm", *Applied Soft Computing*, vol.11: pp.1556–1564, 2011.
[19] Bilal Alatas, "Chaotic harmony search algorithms ", *Applied Mathematics and Computation*, vol.216 : pp.2687–2699 , 2010.
[20] M. Mahdavi,M. Fesanghary,E. Damangir, "An improved harmony search algorithm for solving optimization problems", *Applied Mathematics and Computation,* vol.188: pp.1567–1579 , 2007.
[21] Jing Chen, Quan-ke Pan, Jun-qing Li, "Harmony search algorithm with dynamic control parameters", *Applied Mathematics and Computation*, vol. 219: pp.592.–604 ,2012.
[22] Zong Woo Geem,"Improved Harmony Search from Ensemble of Music Players", *KES 2006, Bournemouth, UK, Part I, LNAI 4251*: pp.86–93,2006.
[23] Zong Woo Geem,"Optimal cost design of water distribution networks using harmony search", *Engineering Optimization*, vol. 38, no. 3, pp.259-277, 2006.

Ming Huang Male, Professor of Dalian Jiaotong University, his research interest is software engineering.

Shujie Guo Male, PhD candidate, his research interest is software engineering.

Xu Liang Female, Professor of Dalian Jiaotong University, her research interest is information management.

Xuan Jiao Female, PhD candidate, her research interest is s business administration.