The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.48 e-ISSN: 2251-7545

GUI Test Coverage Analysis using NSGA II

Abdul Rauf
CCIS,Al-Imam Muhammad Bin Saud University, Riyadh, KSA. 11432
rauf.malik@ccis.imamu.edu.sa

Eisa Aleisa
CCIS,Al-Imam Muhammad Bin Saud University, Riyadh, KSA. 11432
eisa@ccis.imamu.edu.sa

Imam Bakhsh
CCIS,Al-Imam Muhammad Bin Saud University, Riyadh, KSA. 11432
imam.bakhsh@ccis.imamu.edu.sa

Abstract—**Graphical User Interface (GUI) is a mean of interaction between an end user and a software system. Software systems have gained an unprecedented popularity in last twenty years or so and the biggest factor behind this success is Graphical user interface. Software developing companies and teams have always shown a thirst for fully assured high quality software. To fulfill this deep desire of companies, software must go through an intensive testing, but it seems almost impossible to test a GUI application manually due to complexity involve in such effort. Obvious alternative is to go for automated testing. Models or Graphs are being considered as basis for automated GUI testing. Event-flow graph is one of several efforts towards automation of GUI testing. Thorough testing to satisfy the test organization or team's demands is also a terminology, facing lack of consensus among different researchers. Usually test criterion corresponds a "coverage function" that measures how much of the automatically generated optimization parameters satisfies the given test criterion. Our past work has demonstrated that with the help of evolutionary algorithms and event flow representation we can get promising test coverage of GUI applications. Now we are going to extend our previous work and proposing the use of an evolutionary algorithm to gain multiple objectives. These objectives are to gain maximum coverage while keeping number of test cases at minimum side, and the evolutionary algorithm we are going to use for this purpose is Non-dominated Sorting Genetic Algorithm II (NSGA-II).**

Keywords- GUI Testing; Multiobjective optimization; Coverage Criterion; Coverage Analysis; Event Flow; Test Data Generation; NSGA-II; Automation Testing;

I. INTRODUCTION

Software Testing is probably most significant and imperative phase in software development life cycle. An emergent apprehension among organizations developing software is regarding a comprehensive and accurate but speedy software testing process. Traditionally software testing is a challenging activity from all three prospective; time, cost and effort. According to a study [1], software testing is consuming more or less 67% of the entire cost of software development [1]. Keeping this growing concerns involved, automated testing is getting very rapid popularity. Major advantages being offered by automated software testing are speedy test execution process, smooth and repeatable testing process as well as a high coverage of functionality.

Second vital concept of our research relates to Graphical User Interface (GUI). Importance of GUI is also one of the most emergent practical areas in computer science mainly because of ease and flexibility provided by these interfaces [2]. Contrary to its popularity in practice and development, GUI Testing is unable to gain the momentum in research field. A user can access a particular component in a software system by following several itineraries of events. Above mentioned fact depicts the freedom offered by GUI. Large numbers of permutations of events and complex event interactions of GUIs present new challenges both for researchers and practitioners of GUI testing area.

In traditional software testing, the evolutionary algorithms (EAs) have been used to prevail over challenges of manual software testing by introducing different flavors of automation in software testing. Most of the times, two major types of evolutionary algorithms have been used for software testing: Single objective evolutionary algorithms and multiple objective evolutionary algorithms. A general single-objective optimization problem is defined as minimizing (or maximizing) f(x) subject to $g_i(x) \leq 0$, $i = \{1,2,3 \ldots, m\}$, and $h_j(x) = 0$, $j = \{1,2,3 \ldots, p\}$ $x \in \Omega$. A solution minimizes (or maximizes) the scalar f(x) where x is a n-dimensional decision variable vector $x = (x1,x2 \ldots, xn)$ from some universe Ω [4]. Genetic algorithm (GA) belongs to evolutionary algorithms family and works on single objective optimization principle. GA can be used for finding out optimized test suite for GUI testing as well as it can be used for coverage analysis [3]. In case of GUI testing, genetic algorithm searches for optimal test parameter combinations that satisfy a predefined test criterion. This test criterion is represented through a "coverage function" that measures how much of the automatically generated optimization parameters satisfies the given test criterion. In contrast to single objective optimization provided by GA, nondominated sorting genetic algorithm II (NSGA-II) is one of the most prominent multiobjective evolutionary algorithms (MOEAs) used because it relies heavily on its density estimator mechanisms [4]. In this paper we are proposing GUI test automation by NSGA-II based on event flow nature of GUI. NSGA-II will cover the following objectives.

- To minimize the number of event based GUI test cases.
- To maximize the coverage of GUI test cases.

The remainder of the paper is organized as follows: in next Section, we discuss related work in field of software testing, GUI testing and optimization techniques. Section 3 describes

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.48 e-ISSN: 2251-7545

proposed method in detail. Section 4 presents results of experiments related to test case optimization and maximizing coverage, while in section 5, some future directions have been presented and section 6 concludes the paper.

II. Related Work

Most of the techniques used to test GUIs are being extended from techniques that were used to test simple command line interfaces programs earlier. However, most of these extensions are not as successful when they are applied to GUI's as they are in case of software. Although model based techniques have been used frequently for software testing, but models are very expensive to create, and their applicability is limited as well. For these reasons, model based techniques are not being used for GUI testing frequently, but in past few years, efforts have been made for developing different models for GUI testing. Atif M. Memon and his team have worked a lot in automated GUI testing [8, 9]. They have used several types of graph models (e.g., event-flow graphs) to generate specific types of test cases [8, 9]. In [6], the authors have gathered all the models into one scalable event-flow model and sketches algorithms to semi-automatic reverse engineering an application model. Atif M. memon and Xie also created an event-interaction graph (EIG) [6, 7]. Kasik and George [11] have an innovative idea to resemble novice GUI users. White L, Almezen H, Alzeidi N. has developed a technique to address the User-based testing of GUI sequences and their interaction [12]. White L and Almezen H. have also given techniques to generate test cases responsibilities of graphical user interface using complete interaction sequences [13]. In [9, 10, 14, 15], there have been a number of studies that use genetic algorithms (GA's) for software testing. Jones et al proposed a technique to generate test-data for branch coverage using GA [16, 17]. This technique has shown good results with number of small programs. Pargas et al used a GA based on the control dependence graph to search for test data that give good coverage [18]. They used the original test suite developed for the SUT as the seed for the GA. They compare their system to random testing on six small C programs. For the smallest programs, there is no difference, but for the three largest programs, the GA-based method outperforms random testing. Tracey et al presents a framework for test-data generation based on optimization algorithms for structural testing [19]. Tracey has also used a similar technique for functional (black-box) testing. The research by Tracey et al is unique in that they have evaluated their techniques on a real-world safety-critical system [19]. Yongzhong Lu et al presented a new GUI automation test model based on the event-flow graph modeling [20]. In this model, the authors have presented a technique to generate test cases in the daily smoke test based on an improved ACO and a spanning tree is utilized to create test cases in the deep regression test [20].

Wasif Afzal et al [21] have presented a systematic mapping study of a comprehensive review of primary studies in search engine optimization techniques based on functional tests currently offered. This study is an attempt to identify gaps in the application to query optimization techniques based on different types of non-functional testing.

III. Problem Modeling

In this section we are going to discuss some technical terms related to multi objective optimization problems and working of NSGA-II. Also modeling of multi-objective optimization problem to our scenario is presented in this section.

The multiobjective optimization problem can be defined as "a vector of decision variables which satisfies constraints and optimizes a vector function whose elements represent the objective functions. These functions form a mathematical description of performance criteria which is usually to resolve each. Therefore, the term "optimize" means finding such a solution that can give the value of all objective functions acceptable to the manufacturer's decision [5].

In multiobjective optimization problem, we have multiple functions to optimize, so the concept of optimizing function changes. As we have to find a good tradeoff between function values. In our case, we have two objectives and which are inversely proportional to each other; i.e. maximizing one objective results in the minimization of the other objective function. Our objective functions are number of test cases and required coverage. So we have to find a good compromise in between optimization of both objectives. The most commonly accepted term for finding this optimum solution is Pareto optimum. "A solution x ε Ω is said to be Pareto Optimal with respect to (w.r.t.) Ω if and only if (iff) there is no $x' \in \Omega$ for which $v = F(x') = (f_1(x').....f_n(x'))$ dominates $u = F(x) = (f_1(x).....f_n(x))$ [23, 24, 25].

IV. Proposed Method

A GUI is a hierarchical, graphical front-end to a software system that accepts as input user-generated and system-generated events from a fixed set of events and produces deterministic graphical output [3]. A GUI contains graphical objects; each object has a fixed set of properties. At any time during the execution of the GUI, these properties have discrete values, the set of which constitutes the state of the GUI [3].

To test GUI and analyze the coverage, we have proposed a method based upon Multi Objective Genetic Algorithms (MOGAs). For this purpose we have used NSGA-II (A well Known version of MOGA). These objectives are to gain maximum coverage while keeping number of test cases at minimum side, and the evolutionary algorithm.

NSGA is a popular non-domination based genetic algorithm for multi-objective optimization. A modified version, NSGA-II ([3]) was developed, which has a better sorting algorithm , incorporates elitism and no sharing parameter needs to be chosen a priori. To test GUI and analyze the coverage, we have proposed a method based upon NSGA-II.

Working of NSGA-II has been explained with the help of a block diagram in figure 1.

We have divided our proposed system into two major blocks.

- Test data [test cases] generation

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.48 e-ISSN: 2251-7545

- Optimization [minimization] of test paths [cases] using NSGA-II

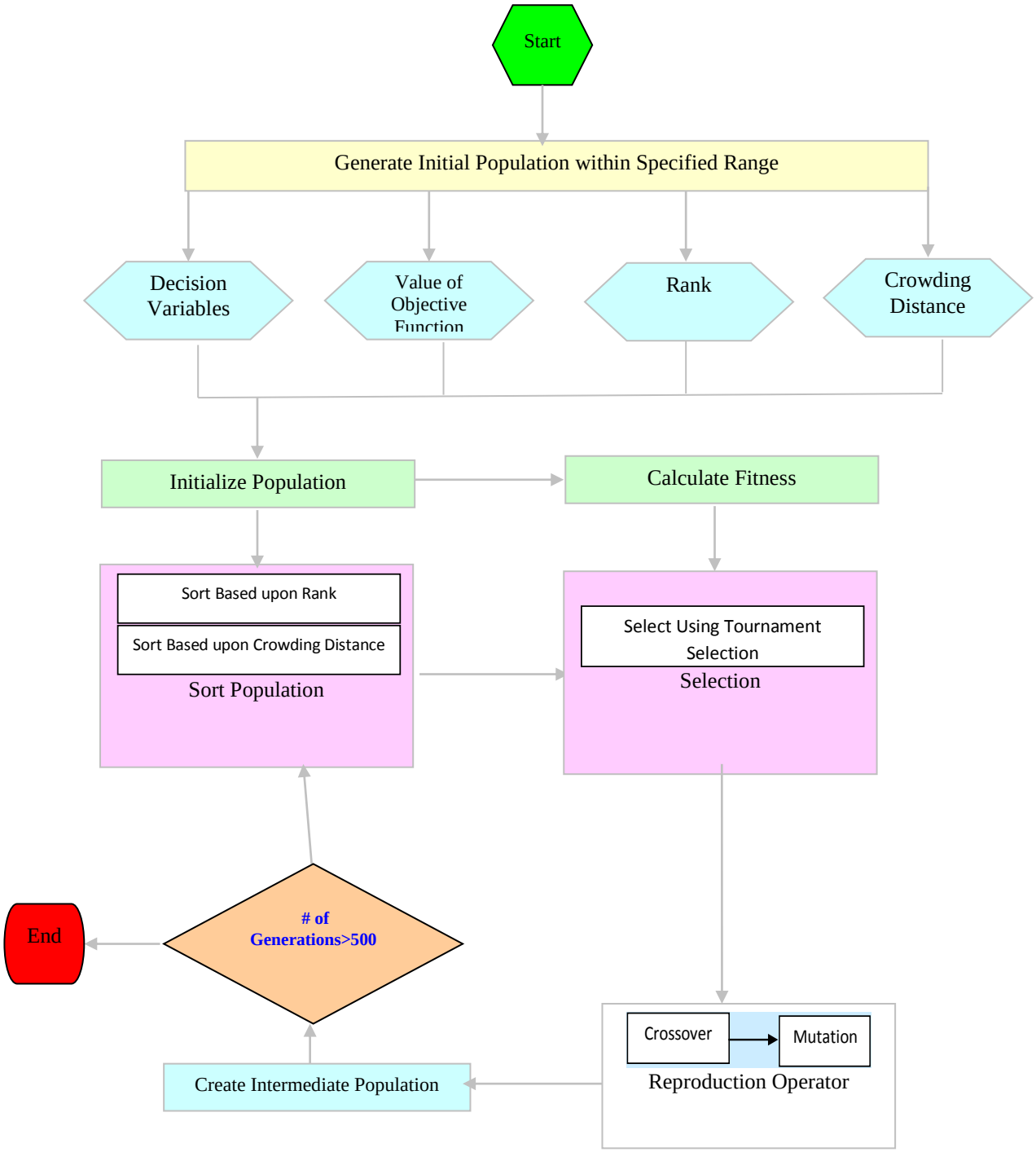

Figure 1. Figure 01: Block Diagram of NSGA-II

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.48 e-ISSN: 2251-7545

A. Test Data Generation

For test data generation, we have used event based techniques. For this purpose we have developed a calculator that receives inputs both from mouse as well as from the keyboard. For every event, there is a unique event ID, as an event occurs, by the help of mouse or a key stroke, respective event ID gets added into event recorder as has been shown in figure 2. After completion of user interaction with the calculator, a sequence of events is formulated, this is passed to next phase for further analysis. Sequence of generated events has been shown in figure 3.

```
1000: CE BUTTON
1001: C Button

2000: 0
2001: 1
2002: 2
2003: 3
2004: 4
2005: 5
2006: 6
2007: 7
2008: 8
2009: 9
2010: .

3000: =
3001: +
3002: -
3003: *
3004: /

4000: Check box for input from mouse and keyboard
4001: Check box for input only from mouse only
```

Figure 2. Figure 02: Event ID's of Calculator Application

```
2001
2002
2007
2008
2009
2006
2004
2002
2001
1000
4000
4001
2000
3000
2008
5002
2001
2002
2005
3000
```

Figure 3. Figure 03: Sequence of Generated Events

B. Optimization of Test Paths using NSGA-II

Genetic algorithms are inspired by Darwin's theory about evolution. Solution to a problem SOLVED BY genetic algorithms is developed. Algorithm starts with a set of solutions (represented by chromosomes) CALLED POPULATION. Solutions from a population sampled and used to form a new population. Following are steps of NSGA-II that we followed for analysis of test path coverage analysis:

1) Initialize the population

Generate random population of n chromosomes. Chromosomes have been formed from the captured events sequences as shown above. Length of our chromosome is the longest path (Longest test case). We have initialized these chromosomes between 1 and maximum length of the test case.

2) Sort the population using non-domination-sort

In this case we have sorted the population using non-domination-sort. This returns two vectors for each individual which are the rank and the crowding distance corresponding to their position in the front they belong. At this stage the rank and the crowding distance for each chromosome is added to the chromosome vector for ease of computation.

3) Start the evolution process

Population is initialized with random values which are within the specified range. Each chromosome consists of the decision variables. Our fitness function is how much test cases have successfully validated?

Accuracy = Test Paths covered by chromosome/ Total number of chromosome

4) For each generation

a) Select the parents which are fit for reproduction
Select two parent chromosomes from a population according to higher fitness.

b) Perform crossover and Mutation operator on the selected parents. We have applied these reproduction operators to increase the coverage efficiency. Also we have generated a random number to find the mutation point in chromosome.

c) Create Intermediate population Intermediate population is the combined population of parents and offsprings of the current generation.

d) Non-domination-sort of intermediate population

e) Perform Selection

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.48 e-ISSN: 2251-7545

5) end

V. RESULTS AND DISCUSSION

Our proposed method resolves the multi-objective problem by using non-dominance based selection. Our technique initially generates multiple solutions and then optimizes the solutions using crowding distance and ranking. The solutions are then evaluated using the pre-defined multiple quality measures. Then non-dominating solutions are selected to build Pareto Front. Table 01 show the coverage achieved against number of generations for which the experimentation was performed.

In the figure 04, graph has been shown for pareto front formed by plotting coverage achieved and number of test cases. Fig 04 shows that while trying to maximize one objective, we are facing a tradeoff in form of minimization in other objective function. Hence we have to find a good compromise in between optimization of both objectives depending upon the requirements and circumstances. Also the results have shown the overall effectiveness and improvement that our proposed technique has achieved in effective coverage analysis. We are in the process of generating further test cases for other applications to further examine the performance of our approach for coverage analysis.

TABLE I. COVERAGE ACHIEVED ACCORDING TO NUMBER OF GENERATIONS

Number of Generations	Coverage Achieved
300	70%
325	73%
350	78%
375	82%
400	85%
425	90%
450	91%
475	91%
500	91%

VI. CONCLUSION AND FUTURE WORK

Graphical User Interface testing has always been considered a critical element of overall testing paradigm for software applications. In this paper, we have proposed a multi objective genetic algorithm based technique for coverage analysis of GUI testing. The technique has been subjected to extensive testing, and the experiments have shown encouraging results. The results have also shown enhanced coverage increase in number of generations. The proposed technique offers an exciting new area of research which can be applied using different other artificial intelligence techniques.

Our aim is to extend this technique in such a way that it is automatically able to generate correct test data for the complete test coverage.

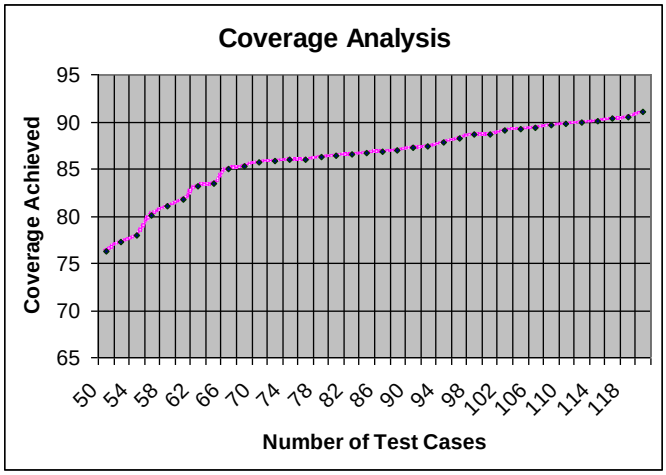

Figure 4. Coverage analysis of proposes technique

REFERENCES

[1]: Hackner, D. R. and Memon, A. M. 2008. Test case generator for GUITAR. In *Companion of the 30th international Conference on Software Engineering* (Leipzig, Germany, May 10 - 18, 2008). ICSE Companion '08. ACM, New York, NY, 959-960.
[2]: Memon, A. M. et al. An event-flow model of GUI-based applications for testing: Research Articles. Softw. Test. Verif. Reliab. 17, 3 (Sep. 2007), 137-157. DOI= http://dx.doi.org/10.1002/stvr.v17:3
[3]: Abdul Rauf et al, "Automated GUI Test Coverage Analysis using GA" 2010 Seventh International Conference on Information Technology (ITNG 2010) to be held 12-14 April 2010 in Las Vegas, Nevada, USA
[4]: Evolutionary Algorithms for Solving Multi-Objective Problems book by Carlos A. Coello Coello Gary B. Lamont David A. Van Veldhuizen
[5]: A. Osyczka. Multicriteria optimization for engineering design. In J. S. Gero, editor, Design Optimization, pages 193–227. Academic Press, 1985.
[6]: Memon, A. M. 2007 et al. An event-flow model of GUI-based applications for testing: Research Articles. Softw. Test. Verif. Reliab. 17, 3 (Sep. 2007), 137-157. DOI= http://dx.doi.org/10.1002/stvr.v17:3
[7]: Qing Xie and Atif M Memon. Using a pilot study to derive a GUI model for automated testing ACM Transactions on Software Engineering and Methodology Volume 18 , Issue 2 (November 2008) Article No. 7 Year of Publication: 2008 ISSN:1049-331X
[8]: MEMON, A. M. 2001 et al. A comprehensive framework for testing graphical user interfaces. Ph.D. dissertation. Department of Computer Science, University of Pittsburgh, Pittsburgh, PA.
[9]: Memon AM, Xie Q. Studying the fault-detection effectiveness of GUI test cases for rapidly evolving software. IEEE Transactions on Software Engineering 2005; 31(10):884–896.

Doi: 10.7321/jscse.v3.n3.48 e-ISSN: 2251-7545

[10]: Atif M. Memon , Mary Lou Soffa , Martha E. Pollack, Coverage criteria for GUI testing, Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering, September 10-14, 2001, Vienna, Austria
[11]: David J. Kasik , Harry G. George, Toward automatic generation of novice user test scripts, Proceedings of the SIGCHI conference on Human factors in computing systems: common ground, p.244-251, April 13-18, 1996, Vancouver, British Columbia, Canada [doi>10.1145/238386.238519]
[12]: White L, Almezen H, Alzeidi N. User-based testing of GUI sequences and their interaction. Proceedings of the International Symposium on Software Reliability Engineering, 8–11 November 2001. IEEE Computer Society Press: Piscataway, NJ, 2001; 54–63.
[13]: White L, Almezen H. Generating test cases for GUI responsibilities using complete interaction sequences. Proceedings of the International Symposium on Software Reliability Engineering, 8–11 October 2000. IEEE Computer Society Press: Piscataway, NJ, 2000; 110–121.
[14]: Memon AM, Pollack ME, Soffa ML. Using a goal-driven approach to generate test cases for GUIs. Proceedings of the 21st International Conference on Software Engineering, May 1999. ACM Press: New York, 1999; 257–266.
[15]: Memon AM, Pollack ME, Soffa ML. Hierarchical GUI test-case generation using automated planning. IEEE Transactions on Software Engineering 2001; 27(2):144–155.
[16]: B.F. Jones, D.E. Eyres, H.H. Sthamer, A strategy for using Genetic Algorithms to automate branch and fault-based testing, The Computer Journal 41(2) (1998) 98-107.
[17]: B.F. Jones, H.H. Sthamer, D.E. Eyers, Automatic structural testing using genetic algorithms, The Software Engineering Journal 11(5) (1996) 299-306.
[18]: R. Pargas, M. J. Harrold, and R. Peck. Test-data generation using genetic algorithms Journal of Software Testing, Verification and Reliability, 9(4):263–282, 1999.
[19]: N. Tracey, J. Clark, K. Mander, J. McDermid, Automated test-data generation for exception conditions, Software Practice and Experience, 30(1) (2000) 61-79
[20]: Yongzhong Lu Danping Yan Songlin Nie Chun Wang, Development of an Improved GUI Automation Test System Based on Event-Flow Graph. International Conference on Computer Science and Software Engineering, Date: 12-14 Dec. 2008
[21]: Afzal, W., Torkar, R., and Feldt, R. 2009. A systematic review of search-based testing for non-functional system properties.Inf. Softw. Technol. 51, 6 (Jun. 2009), 957-976.
[22]: {Ferligoj1992} A. Ferligoj and V. Batagelj, "Direct multicriterion clustering," J. Classification, vol. 9, pp. 43-61, 1992
[23] C. A. Coello Coello. Theoretical and Numerical Constraint-Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art. *Computer Methods in Applied Mechanics and Engineering*, 191(1112):1245–1287, January 2002.
[24] C. A. Coello Coello and G. B. Lamont, editors. *Applications of Multi-Objective Evolutionary Algorithms*. World Scientific, Singapore, 2004. ISBN 981-256-106-4.
[25] D. A. Van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, May 1999.