

EMPIRICAL EVALUATION OF MULTI-OBJECTIVE OPTIMIZATION ALGORITHMS SEARCHING FOR HIGHER ORDER MUTANTS

Quang Vu Nguyen, Lech Madeyski

Faculty of Computer Science and Management, Wrocław University of Technology,

Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland

Email: quang.vu.nguyen@pwr.edu.pl, Lech.Madeyski@pwr.edu.pl

Abstract. First order mutation testing is used to evaluate the quality of a given set of test cases by inserting single changes into the program under test to produce first order mutants (FOMs) of the original program, and then checking whether tests are good enough to detect the artificially injected defects or not. However mutation testing is not yet widely used due to the problems of a large number of generated mutants and limited realism of introduced changes that not necessarily reflect real software defects. Furthermore, many of generated mutants are equivalent, i.e., keep the program semantics unchanged and thus cannot be detected by any test suite. Higher order mutation testing has been coined as a promising solution for overcoming the above mentioned limitations of first order mutation testing. In particular, finding strongly subsuming higher order mutants (SSHOMs), which are able to replace all of its constituent FOMs without scarifying test effectiveness and, simultaneously, able to reflect complex, real defects, which require more than one change to correct them, is considered an important research challenge we focus on. The contribution of this paper embraces a new, extended classification of higher order mutants (HOMs) to cover all cases of generated HOMs, and fitness functions and empirical comparison of four different multi-objective optimization algorithms to generate, evaluate HOMs as well as search for valuable high quality and reasonable HOMs (strongly subsuming and coupled HOMs) and 10 other ten types of HOMs. The main goal of this paper is to assert the effect of applying multi-objective optimization algorithms in the area of higher order mutation testing, and at the same time to assert the correctness of the proposed HOMs classification, objectives and fitness functions. Our experimental results show that the total number of generated HOMs is smaller (about 70%) in comparison to FOMs, while the mean ratio of reasonable HOMs (subsuming HOMs) to all found HOMs is over 56% and the mean ratio of high quality and reasonable HOMs (strongly subsuming and coupled HOMs) to all found reasonable HOMs (subsuming HOMs) is fairly high (around 8.74%).

Keywords: Mutation Testing; Higher Order Mutation; Higher Order Mutants; Strongly Subsuming;

Multi-objective optimization algorithm; High quality; Reasonable.

*This is an Accepted Manuscript of an article published on 09 Feb 2016 by Taylor & Francis Group in **Cybernetic and Systems**, Vol. 47, Iss. 1-2 (2016), p. 48-68, available online: <http://dx.doi.org/10.1080/01969722.2016.1128763>*

INTRODUCTION

There are many optimization problems, which have more than one objective function and the objective functions are conflicting, to some extent, preventing simultaneously the simple optimization of each objective. Multi-objective optimization algorithms, which have been used for solving optimization problems and making the decisions that satisfy multiple optimized objectives, are successful methods. In this paper, we investigate applying multi-objective optimization algorithms to search for HOMs, which are able to improve the quality of first order mutation testing.

The first order mutation testing, originally proposed in 1970s by DeMillo et al. (1978) and Hamlet (1977), has been introduced as a technique to assess the quality of test cases. Mutation testing is used to evaluate the fault detection capability of the test suits by inserting changes into the original program to generate mutations, and then checking whether the given set of test cases is good enough to detect the difference between original program and its mutants or not. Mutants are the different versions of an original program generated by inserting, via a mutation operator, only one semantic change (or fault) into the original program. If a test case distinguishes between the mutant and the original program, it is said to kill the mutant. In other words, the mutant is killed by the test case. Conversely, a mutant is said to be “alive” if no test case detects the injected fault. One of the limitations of the first order mutation testing is a large number of generated mutants, while most of them are simple and easy to detect and do not denote realistic faults (Nguyen and Madeyski 2014, Purushothaman and Perry 2005). Subsequent serious problem of mutation testing is the generation of many, useless equivalent mutants (Harman et al. 2010, Madeyski et al. 2014, Nguyen and Madeyski 2014, Papadakis et al. 2015). The main goal of higher order mutation testing, which was first proposed in 2009 by Jia and Harman, is to generate higher order mutants (HOMs) that can be used to improve the effectiveness of mutation testing.

HOMs are the mutants generated by combining two or more first order mutants. Jia and Harman (2009) (also Harman et al. (2010)) introduced six types of generated HOMs on the basis of two rules:

Rule 1: If a HOM is harder to kill than its constituent first order mutants (FOMs) then it is called a “*Subsuming HOM*”, otherwise it is a “*Non-Subsuming HOM*”. Subsuming HOMs can be divided into two groups: “*Strongly Subsuming HOM*” if a set of test cases (TCs) which kills HOM is a subset of the intersection of the sets of TCs which kill FOMs, otherwise it is a “*Weakly Subsuming HOM*”.

Rule 2: A HOM is called “*Coupled HOM*” if a set of TCs that kills its constituent FOMs also contains cases that kill HOM. Otherwise, it is called “*De-coupled HOM*”.

Assume that h is a HOM constructed from FOMs f_1, \dots, f_n , T is the set of all given test cases, T_h is the subset of T that kills the HOM h , while T_1, \dots, T_n are the subsets of T that kill the constituent FOMs f_1, \dots, f_n respectively. On the basis of the aforementioned rules, Jia and Harman (2009) defined six types of HOMs:

(1) Strongly Subsuming and Coupled:

$$T_h \subset \bigcap_{i=1}^n T_i \text{ and } T_h \neq \emptyset$$

(2) Weakly Subsuming and Coupled

$$|T_h| < \left| \bigcup_{i=1}^n T_i \right|, T_h \neq \emptyset \text{ and } T_h \cap \bigcup_{i=1}^n T_i \neq \emptyset$$

(3) Weakly Subsuming and Decoupled

$$|T_h| < \left| \bigcup_{i=1}^n T_i \right|, T_h \neq \emptyset \text{ and } T_h \cap \bigcup_{i=1}^n T_i = \emptyset$$

(4) Non-Subsuming and De-coupled

$$|T_h| \geq \left| \bigcup_{i=1}^n T_i \right|, T_h \neq \emptyset \text{ and } T_h \cap \bigcup_{i=1}^n T_i \neq \emptyset$$

(5) Non-Subsuming and Coupled

$$|T_h| \geq \left| \bigcup_{i=1}^n T_i \right| \text{ (Useless)}$$

(6) Equivalent mutants

$$T_h = \emptyset$$

Strongly Subsuming and Coupled HOM is harder to kill than any constituent FOMs and only be killed by subsets of the intersection of sets of test cases that kill each constituent FOM. As suggested by Jia and Harman (2009), Strongly Subsuming and Coupled HOM can be used to replace all of its constituent FOMs without loss of test effectiveness. Therefore, finding Strongly Subsuming and Coupled HOMs can help us improve the effectiveness of mutation testing process.

Still there is little research in the field of applying multi-objective optimization algorithms to search for Strongly Subsuming and Coupled HOMs, although this is a promising approach. There are three papers by Harman et al. (2010) and Langdon et al. (2009, 2010), which focused on multi-objective higher order mutation testing with genetic programming. They used only one multi-objective optimization algorithm, NSGAII, to search for higher order mutants which are both hard to kill and realistic.

The contribution of this paper is to propose a new classification of HOMs to cover all of available cases of generated HOMs. We also propose new objectives and fitness functions as well as perform an empirical evaluation of four different multi-objective optimization algorithms for constructing HOMs and finding valuable strongly subsuming and coupled HOMs (named “high quality and reasonable HOMs” in our HOMs classification). We apply four multi-objective optimization algorithms - eMOEA (Epsilon-MOEA), NSGAII, eNSGAII (Epsilon-NSGAII), NSGAIII - to search for valuable high quality and reasonable HOMs for three given Java open source projects. We not only compare

the effectiveness of the four proposed algorithms in term of their ability to find high quality and reasonable HOMs but also analyze the number of categories of HOMs generated.

The rest of the paper is organized as follows. The second section presents the landscape of the reported research on mutation testing and higher order mutation testing. The third section includes our approach for classification and identifying HOMs based on new, proposed by us objectives and fitness function. The fourth section presents the experimental goals, the proposed multi-objective optimization algorithms and implementation details. The fifth section shows results of the empirical evaluation required to answer the posed research questions. The sixth section presents the discussion of our approach and results, while threats to validity are discussed in seventh section. The last section presents conclusions and proposition of future works.

BACKGROUND

First order mutation testing (traditional mutation testing (MT)) (DeMillo et al. 1978, Hamlet 1977) is a highly automated and useful technique for evaluating the fault-finding effectiveness (Madeyski 2007, 2008, 2010b) of tests and many researchers are now involved in mutation testing research. However, there are three main challenges (Harman et al. 2010, Madeyski et al. 2014, Nguyen and Madeyski 2014): a large number of mutants (this also leads to a very high computational cost); realism of generated mutants; and equivalent mutant problem (Madeyski et al. 2014, Papadakis et al. 2015). That is also one of the reasons why mutation testing is not yet widely adopted in practice. A lot of approaches have been proposed for overcoming the MT's problems (Nguyen and Madeyski 2014) including second order mutation testing (Madeyski et al. 2014, Papadakis and Malevris 2010, Polo et al. 2008, Kintis et al. 2010) or higher order mutation testing (Jia and Harman 2008, 2009, Harman et al. 2010, Langdon et al. 2010) in general.

Higher order mutation testing is an idea presented by Jia and Harman (2009) and in a manifesto by Harman et al. (2010). This promising idea offers solutions to overcome the limitations of traditional mutation testing. Mutants can be classified into two types: First Order Mutants (FOMs) and Higher Order Mutants (HOMs). The first are used in traditional mutation testing and generated by applying mutation operators only once in each mutant. The second are used in higher order mutation testing and constructed by inserting two or more changes per mutant.

According to research results of second order mutation testing (Madeyski et al. 2014, Papadakis and Malevris 2010, Kintis et al. 2010, Polo et al. 2008), not only about 50% mutants were reduced without loss of effectiveness of testing (Madeyski et al. 2014, Papadakis and Malevris 2010, Polo et al. 2008), but also the number of equivalent mutants can be reduced (i.e., the reduction in the mean percentage of equivalent mutants passes from about 18.66% of total of FOMs to about 5% of total of HOMs (Polo et al. 2008) or the second order mutation testing significantly reduced the number of equivalent mutants in comparison to the first order mutation, while the size of the effect was medium (Madeyski et

al. 2014)) and generated second order mutants (SOMs) can be harder to kill than first order mutants (Madeyski et al. 2014, Papadakis and Malevris 2010, Kintis et al. 2010, Polo et al. 2008).

In 2010, Jia and Harman introduced some approaches to find Subsuming HOMs, Strongly Subsuming HOMs and other types of HOMs (see first Section) by using the following algorithms: Greedy, Genetic and Hill-Climbing. Their experiments showed approximately 15% of all found Subsuming HOMs were Strongly Subsuming HOMs and they also indicated that finding Strongly Subsuming HOMs may not be too difficult.

Based on the claim of Purushothaman and Perry (2005) which indicated that a modification made to fix one real fault needs several source code changes, Langdon et al. (2009) believed that most of the generated FOMs are simple and do not denote realistic faults. In order to produce better mutants, they suggested inserting “semantically close” faults instead of inserting “syntactically close” faults to the original program under test. They applied NSGA-II, a multi-objective optimization algorithm, with genetic programming in the area of higher order mutation testing and the results demonstrated that this approach is able to find higher order mutants that represent more realistic complex faults and harder to kill.

In this paper, we propose not only a new classification of HOMs but also new objectives and fitness functions. And based on that, our research focuses on applying and comparing the effectiveness of four different multi-objective optimization algorithms to search for valuable high quality and reasonable HOMs with our HOMs classification as well as our objectives and fitness functions. Our main goal is to assert the effect of searching high quality and reasonable HOMs applying multi-objective optimization algorithms, and at the same time to assert the correctness of the proposed HOMs classification, objectives and fitness functions in the next Section.

AN APPROACH FOR HOMs CLASSIFICATION, OBJECTIVES AND FITNESS FUNCTIONS

Our investigation is to propose an approach for constructing HOMs in order to reduce the number of generated mutants as well as to increase realism of mutants. The constructed HOMs should be more realistic (reflect real faults) and harder to kill than any constituent FOMs. It means that the number of TCs, which can kill HOMs, is as small as possible. Based on the idea “number of TCs which can kill HOMs is as small as possible”, we propose a HOMs classification, objectives functions and fitness functions which we will apply to the different multi-objective optimization algorithms to identify and search for HOMs, especially strongly subsuming and coupled HOMs (named “high quality and reasonable HOMs” in our HOMs classification). Each high quality and reasonable HOM represents an optimal solution that we want to find out.

HOMs classification

The idea of new HOMs classification approach is based on the combination of a set of test cases, which kill HOM, and sets of TCs, which kill its constituent FOMs. For the sake of simplicity, we use a second order mutant to illustrate our method for HOMs classification. The notations are explained below (See Figure 1):

H: a HOM, constructed from FOMs: F_1 and F_2

T: The given set of test cases

$T_{F1} \subset T$: Set of test cases that kill FOM1

$T_{F2} \subset T$: Set of test cases that kill FOM2

$T_H \subset T$: Set of test cases that kill H

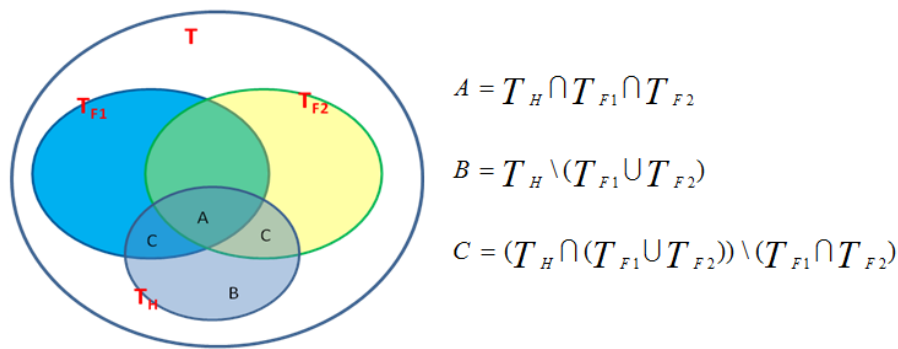


Figure 1. The combination of sets of TCs

$A \subset T_H$: Set of test cases that can kill H and all its constituent FOMs.

$B \subset T_H$: Set of test cases that kill H but cannot kill any its constituent FOMs.

$C \subset T_H$: Set of test cases that kill H and can kill FOM1 or FOM2.

In the proposed classification of higher order mutants of Harman et al. (2009, 2010), they mentioned “Subsuming HOM” with meaning the set of TCs which kills HOM is smaller than the set of TCs which kills constituent FOMs, in other words HOM is harder to kill than FOMs. By contrast, “Non-Subsuming HOM” is easier to kill than FOMs. They also defined that, if a set of TCs that kills its constituent FOMs also contains cases that kill HOM then the HOM is called “Coupled HOM”, otherwise it is called “De-coupled HOM” (Jia and Harman 2009, Harman et al. 2010). Based on that, they classified HOMs into 6 groups (see First Section). However, they did not mention the cases as follows:

Case 1: The set of TCs, which can kill the HOM (T_H), is a subset of the set of TCs, which can kill FOM1 or FOM2.

It means that, B is an empty set, T_H includes only A and C.

Case 2: The set of TCs, which can kill the HOM, has some TCs that belong to the set of TCs, which can kill FOM1 or FOM2 but not both, while other TCs do not belong to the aforementioned set. It means that A is an empty set, T_H includes only B and C.

We extend the proposed classification of higher order mutants of Harman et al. (2009, 2010) by proposing our new rules in order to cover all of available cases of generated HOMs. The categories of HOMs are named on a basis of the new rules below (Rule 3, 4, 5, 6 and 7), and are illustrated Table 1. The ^(*) notation is used in this paper in order to show the names that were defined by Harman et al. (2009, 2010).

Rule 3: If set of test cases $A \subset T_H$ (see Figure 1), which can kill a HOM and all its constituent FOMs, is not empty then HOM is called a “quality HOM”, otherwise it is a “non-quality HOM”. In case $A \subset T_H$ is a not empty set while both B and C are empty, HOM is called a “high quality HOM”.

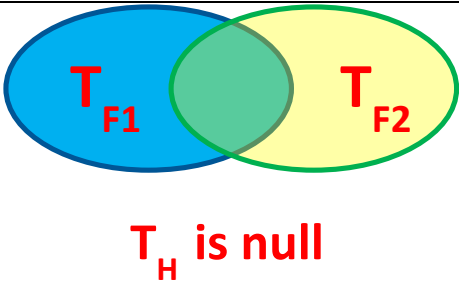
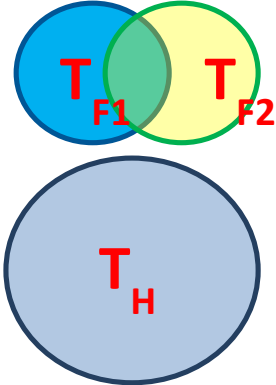
Rule 4: If a HOM is harder to kill than its constituent FOMs then it is a “reasonable HOM” (subsuming HOM^(*)), otherwise it is a “un-reasonable HOM” (non-subsuming HOM^(*)).

Rule 5: A HOM, which is killed by a set of TCs that can kill FOM1 or FOM2, is called “HOM With Old Test Cases”. In case the aforementioned set of TCs is a subset of A (see Figure 1), it is “high quality and reasonable HOM” (Strongly Subsuming and Coupled HOM^(*)).

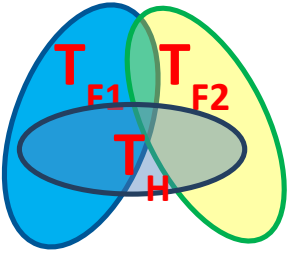
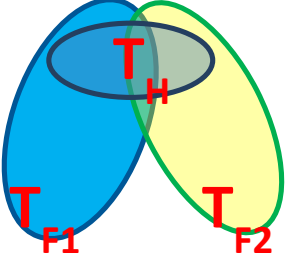
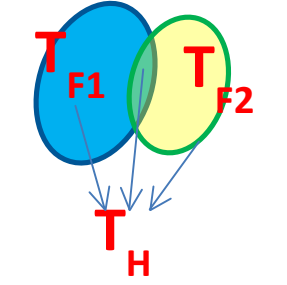
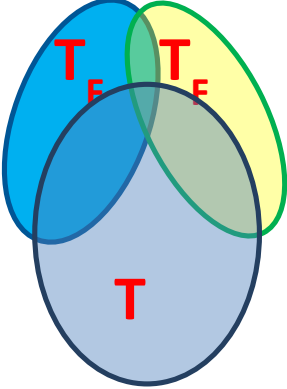
Rule 6: A HOM, which is killed by a set of TCs that cannot kill any their constituent FOMs (this set of TCs $\subset B$, see Figure 1), is called “HOM With New Test Cases”.

Rule 7: A HOM, which is killed by a set of TCs which has some TCs $\subset B$ and some others TCs $\subset A$ or $\subset C$ (see Figure 1), is called “HOM With Mixed Test Cases”.

Table 1. Eleven categories of HOMs based on the combination of sets of testcases.

Case	Illustration	HOM is
H1		Equivalent Mutant ^(*) (T_H is null)
H2		Non-Quality, Un-Reasonable and With New TCs.

H3		Non-Quality, Un-Reasonable and With Mixed TCs.
H4		Non-Quality, Reasonable and With New TCs.
H5		Non-Quality, Reasonable and With Mixed TCs.
H6	$T_H = (T_{F1} \cup T_{F2}) \setminus (T_{F1} \cap T_{F2})$	Non-Quality, Reasonable and With Old TCs.
H7		High quality and Reasonable (Strongly Subsuming and Coupled ^(*))

H8		Quality, Reasonable and With Mixed TCs.
H9		Quality, Reasonable and With Old TCs.
H10	 $T_H = T_{F1} \cup T_{F2}$	Quality, Un-Reasonable and With Old TCs.
H11		Quality, Un-Reasonable and With Mixed TCs.

Objectives and fitness functions

As we mentioned above, we want to search for the HOMs, especially High quality and Reasonable HOMs (strongly subsuming and coupled HOMs^(*)), which are killed by as small as possible set of TCs. Therefore, we focus on the set of TCs, which can kill HOMs and its subsets. The Figure 1 shows that the set of TCs that kills a HOM can be divided into 3 subsets:

- The first is the subset that can kill HOM and all its constituent FOMs (subset A)
- The second is the subset that kills HOM and can kill FOM1 or FOM2 (subset C)
- The third is the subset that only kills HOM and cannot kill any FOMs (subset B)

HOM is called a High quality and Reasonable HOM when the subsets B, C are simultaneously empty and subset A is not empty. In this case, the number of TCs (subset A) that can kill High quality and Reasonable HOM is as small as possible. If all of 3 subsets are empty, HOM is equivalent HOM. If A and C are empty, B is not empty, we call “HOM With New Test Cases”. By contrast, if B is empty but A and C are not empty, we call “HOM With Old Test Cases”. If B is not empty and at least one of A or C is not empty, we call “HOM With Mixed Test Cases”.

From that, we propose objectives and fitness functions (see equations below) to apply multi-objective optimization algorithms for searching high quality and reasonable HOMs (as well as other types of HOMs) as follows:

Objective 1: Minimize the number of TCs that kill HOM and also kill all its constituent FOMs (The fitness function is fitness(OB1) in Equation 1).

Objective 2: Minimize the number of TCs that kill HOM but cannot kill any their constituent FOMs (The fitness function is fitness(OB2) in Equation 2).

Objective 3: Minimize the number of TCs that kill HOM and can kill FOM1 or FOM2 (The fitness function is fitness(OB3) in Equation 3).

$$fitness(OB1) = \frac{\#(T_H \cap T_{F1} \cap T_{F2})}{\#T_H} \quad (1)$$

$$fitness(OB2) = \frac{\#(T_H \setminus (T_{F1} \cup T_{F2}))}{\#T_H} \quad (2)$$

$$fitness(OB3) = \frac{\#((T_H \cap (T_{F1} \cup T_{F2})) \setminus (T_{F1} \cap T_{F2}))}{\#T_H} \quad (3)$$

$$fitness(H) = \frac{\#T_H}{\#(T_{F1} \cup T_{F2})} \quad (4)$$

The values of fitness(OB1), fitness(OB2) and fitness(OB3) lie between 0 and 1. In addition, we also propose the fitness(H) function (Equation 4) for the objective “HOM is harder to kill than its constituent FOMs”. The fitness(H) is used to evaluate a HOM whether it is a “reasonable HOM” or “un-reasonable HOM”. The value of fitness(H) can be divided into 3 groups: If fitness(H)=0, HOM is a equivalent mutants; If 0<fitness(H)<1, HOM is a reasonable HOM; If fitness(H)>=1, HOM is a un-reasonable HOM. Generated HOMs are identified on a basis of the values as shown in Table 2. V1, V2, V3 and V4 are the values of fitness(OB1), fitness(OB2), fitness(OB3) and fitness(H) respectively.

Table 2. Identify HOMs based on the values of fitness functions.

Case	V1	V2	V3	V4	HOM is
H1	0	0	0	0	Equivalent Mutant ^(*) (T_H is null)
H2	0	1	0	≥ 1	Non-Quality, Un-Reasonable and With New TCs.
H3	0	$0 < V2 < 1$	$0 < V3 < 1$	≥ 1	Non-Quality, Un-Reasonable and With Mixed TCs.
H4	0	1	0	$0 < V4 < 1$	Non-Quality, Reasonable and With New TCs.
H5	0	$0 < V2 < 1$	$0 < V3 < 1$	$0 < V4 < 1$	Non-Quality, Reasonable and With Mixed TCs.
H6	0	0	1	$0 < V4 < 1$	Non-Quality, Reasonable and With Old TCs.
H7	$0 < V1 \leq 1$	0	0	$0 < V4 < 1$	High quality and Reasonable (Strongly Subsuming and Coupled^(*))
H8	$0 < V1 \leq 1$	$0 < V2 < 1$	$0 \leq V3 < 1$	$0 < V4 < 1$	Quality, Reasonable and With Mixed TCs.
H9	$0 < V1 \leq 1$	0	$0 < V3 < 1$	$0 < V4 < 1$	Quality, Reasonable and With Old TCs.
H10	$0 < V1 \leq 1$	0	$0 < V3 < 1$	1	Quality, Un-Reasonable and With Old TCs.
H11	$0 < V1 \leq 1$	$0 < V2 < 1$	$0 \leq V3 < 1$	≥ 1	Quality, Un-Reasonable and With Mixed TCs.

EXPERIMENT PLANNING AND EXECUTION

Goals

We pose the following research questions (RQ) that the study will answer:

RQ1: How popular are mutants in the identified mutant categories (H1-H11)?

In this paper, we introduce a new HOMs classification and applied multi-objective optimization algorithms to generate higher order mutants. The goal of RQ1 is to investigate the number (distribution) of HOMs in the identified mutant categories.

RQ2: What are the proportions of “not good” HOMs that were generated?

“Not good” HOMs mean Non-quality and Un-reasonable HOMs (H2 and H3), which are Non-quality HOMs and easier to kill than their constituent FOMs. This question allows us to evaluate the proportions of Non-quality and Un-reasonable HOMs (H2 and H3), which were generated.

RQ3: Which multi-objective optimization algorithms are better suitable for searching high quality and reasonable HOMs and to avoid equivalent mutants?

We use RQ3 to evaluate the ability of proposed four multi-objective optimization algorithms in term of not only the number of generated high quality and reasonable HOMs but also the number of generated equivalent mutants for each algorithm.

Multi-objectives optimization algorithms

We perform an empirical evaluation of multi-objective optimization algorithms, guided by the aforementioned fitness functions, searching for high quality and reasonable HOMs based on the proposed HOMs classification.

NSGA-II is the second version of the Non-dominated Sorting Genetic Algorithm that was proposed by Deb et al. (2002) for solving non-convex and non-smooth single and multi-objective optimization problems. Its main features are: it uses an elitist principle; it emphasizes non-dominated solutions; and it uses an explicit diversity preserving mechanism. NSGA-III is the extension of NSGA-II which is based on the supply of a set of reference points and demonstrated its working in three to 15-objective optimization problems (Deb et al. 2014). The ϵ NSGA-II (ϵ NSGA_II) extends NSGA-II's concepts by adding ϵ -dominance, adaptive population sizing, and self-termination to minimize the need for parameter calibration. ϵ -dominance is a concept where by a user is able to specify the precision with which he wants to obtain the Pareto-optimal solutions to a multi-objective problem, in essence giving him the ability to assign a relative importance to each objective (Adekanmbi et al. 2014, Kollat and Reed 2005). The ϵ MOEA (ϵ MOEA) is a steady state multi-objective evolutionary algorithm that co-evolves both an evolutionary algorithm population and an archive population by randomly mating individuals from the population and the archive to generate new solutions (Kollat and Reed 2005, Deb et al. 2003).

Supporting tool

Judy (Madeyski et al. 2014, Madeyski and Radyk 2010) is a mutation testing tool for Java. It supports large set of mutation operators, as well as HOM generation, HOM execution and mutation analysis. The list of some mutation operators available in Judy (Madeyski et al. 2014, Madeyski and Radyk 2010) is presented in Table 3.

Table 3. Java mutation operators available in Judy

AIR	AIR_Add	Replaces basic binary arithmetic instructions with ADD
	AIR_Div	Replaces basic binary arithmetic instructions with DIV
	AIR_LeftOperand	Replaces basic binary arithmetic instructions with their left operands
	AIR_Mul	Replaces basic binary arithmetic instructions with MUL
	AIR_Rem	Replaces basic binary arithmetic instructions with REM
	AIR_RightOperand	Replaces basic binary arithmetic instructions with their right operands
	AIR_Sub	Replaces basic binary arithmetic instructions with SUB
JIR	JIR_Ifeq	Replaces jump instructions with IFEQ (IF_ICMPEQ, IF_ACMPEQ)
	JIR>Ifge	Replaces jump instructions with IFGE (IFICMPGE)

	JIR_Ifgt	Replaces jump instructions with IFGT (IF_ICMPGT)
	JIR_Ifle	Replaces jump instructions with IFLE (IF_ICMPLE)
	JIR_Iflt	Replaces jump instructions with IFLT (IF_ICMPLT)
	JIR_Ijne	Replaces jump instructions with IFNE (IF_ICMPNE, IF_ACMPLNE)
	JIR_Ifnul	Replaces jump instruction IFNULL with IFNONNULL and vice-versa
LIR	LIR_And	Replaces binary logical instructions with AND
	LIR_LeftOperand	Replaces binary logical instructions with their left operands
	LIR_OrReplaces	Replace binary logical instructions with OR
	LIR_RightOperand	Replaces binary logical instructions with their right operands
	LIR_Xor	Replaces binary logical instructions with XOR
SIR	SIR_LeftOperand	Replaces shift instructions with their left operands
	SIR_Sh1	Replaces shift instructions with SHL
	SIR_Shr	Replaces shift instructions with SHR
	SIR_Ushr	Replaces shift instructions with USHR
Inheritance	IOD	Deletes overriding method
	IOP	Relocates calls to overridden method
	IOR	Renames overridden method
	IPC	Deletes super constructor call
	ISD	Deletes super keyword before fields and methods calls
	ISI	Inserts super keyword before fields and methods calls
Polymorphism	OAC	Changes order or number of arguments in method invocations
	OMD	Deletes overloading method declarations, one at a time
	OMR	Changes overloading method
	PLD	Changes local variable type to super class of original type
	PNC	Calls new with child class type
	PPD	Changes parameter type to super class of original type
	PRV	Changes operands of reference assignment
Java-Specific	EAM	Changes an access or method name to other compatible access or method names
Features	EMM	Changes a modifier method name to other compatible modifier method names
	EOA	Replaces reference assignment with content assignment (clone) and vice-versa
	EOC	Replaces reference comparison with content comparison (equals) and vice-versa
	JDC	Deletes the implemented default constructor
	JID	Deletes field initialization
	JTD	Deletes this keyword when field has the same name as parameter
	JTI	Inserts this keyword when field has the same name as parameter
Jumble-Based	Arithmetics	Mutates arithmetic instructions
	Jumps	Mutates conditional instructions
	Returns	Mutates return values

Increments	Mutates increments
------------	--------------------

Software under test (SUT)

In our empirical study, we use three real-world, open source projects (see Table 4). There is one small project and two larger ones in term of lines of code. Table 4 shows the projects selected for the experiment along with their number of classes (NOC), lines of code (LOC) and number of given test cases.

Table 4. Software under test

Project	NOC	LOC	# Test cases
BeanBin (http://beanbin.sourceforge.net)	72	5925	68
Barbecue (http://barbecue.sourceforge.net)	57	23996	190
JWBF (http://jwbfs.sourceforge.net)	51	13572	305

BeanBin is a tool to make persisting EJB (Enterprise JavaBeans) 3.0 entity beans easier.

Barbecue is a library that provides the means to create barcodes for Java applications.

JWBF (Java Wiki Bot Framework) is a library to maintain Wikis like Wikipedia based on MediaWiki and provides methods to connect, modify and read collections of articles, to help created wiki bot.

Approach

We use Judy mutation testing tool for Java (Madeyski et al. 2014, Madeyski and Radyk 2010) to generate and evaluate FOMs and HOMs with the same full set of mutation operators of Judy (see Table 3) for the three software projects under test. The four applied multi-objective optimization algorithms for producing HOMs base on the MOEA Framework (<http://www.moeaframework.org>). We set out the experimental procedure as follows:

```

for each software under test do
    - generate all possible FOMs by applying the set of Judy mutation
      operators
    - set objectives and fitness functions
for each multi-objective optimization algorithm do
    - set populationSize =100
    - set maxMutationOrder =15
    - from set of FOMs, generate, evaluate HOMs, find as many high
      quality and reasonable HOMs as possible, guided by objective
      functions and fitness function

```

- count the percentage of categories of HOMs
- count the percentage of Non-Quality and Un-Reasonable HOMs
- count the percentage of high quality and reasonable HOMs within the reasonable HOMs

end

end

RESULTS AND ANALYSIS

Differences in the source code of the three SUT projects resulted in some variations in mutation operators finally used to generate HOMs (see Table 5).

Table 5. Mutation operators used

Project	Operators
BeanBin	JIR_Ifle;JIR_Ifeq;JIR_Iflt;JIR_Ifne;JIR_Ifge;JIR_Iflt;JIR_Ifnull; AIR_Div;AIR_LeftOperand;AIR_RightOperand;AIR_Rem;AIR_Mul; EAM;EOC;PLD;CCE;LSF;REV;ISI;JTD;JTI;OAC;DUL;
Barbecue	JIR_Ifge;JIR_Ifeq;JIR_Iflt;JIR_Ifne;JIR_Iflt;JIR_Ifle;JIR_Ifnull; AIR_Add;AIR_Div;AIR_Sub;AIR_LeftOperand;AIR_RightOperand; AIR_Mul;AIR_Rem;JTI;JTD;JID;JDC;EAM;OAC;EOC;FBD;IPC;EGE;EMM;CCE;LSF;REV;
JWBF	JIR_Ifge;JIR_Ifeq;JIR_Iflt;JIR_Ifne;JIR_Iflt;JIR_Ifle; EAM;EOA;JTD;JTI;JID;PRV;OAC;PLD;

The mean value of mutation score indicator (MSI) for each project under test after higher order mutation testing is presented in Table 6. MSI, is the ratio of killed mutants to all generated mutants (Madeyski 2007, 2008, 2010b, Madeyski et al. 2014, Madeyski and Radyk 2010). MSI is a quantitative measure of the quality of test cases. It is different from mutation score (MS) that was defined as the ratio of killed mutants to difference of all generated mutants and equivalent mutants (DeMillo et al. 1978, Hamlet 1977). JWBF is a project with the highest MSI. It means that JWBF is a project, which has higher quality set of test cases than the other analyzed SUTs. While the numbers of live mutants of Barbecue and BeanBin projects are about 30% and 57%. This leads the given sets of test cases of Barbecue and BeanBin projects need to improve the quality in term of fault detection.

Table 6. The mean value of MSI for each project under test (%)

Project	Barbecue	BeanBin	JWBF
---------	----------	---------	------

MSI	69.24	43.09	92.28
------------	-------	-------	-------

Answer to RQ1

Table 7 shows the numbers of generated FOMs of projects under test (PUTs), while Table 8 presents the number of total generated HOMs as well as numbers of HOMs in each category (defined in third Section) in different PUTs using multi-objective optimization algorithms with the proposed objectives and fitness functions.

Table 7. The mean number of generated FOMs

Project	Barbecue	BeanBin	JWBF
Number of FOMs	3084	1330	1482

Table 8. Number of generated HOMs of 11 categories

Pro.	eMOEA			NSGAII			eNSGAII			NSGAIII		
	Bar	Bean	JWBF	Bar	Bean	JWBF	Bar	Bean	JWBF	Bar	Bean	JWBF
HOMs	886	424	216	891	402	238	379	338	123	887	398	230
H1	274	250	19	251	259	13	129	176	11	272	227	20
H2	0	1	0	1	0	0	0	0	0	2	0	0
H3	69	6	3	90	7	6	31	2	2	88	3	4
H4	11	55	0	18	38	0	0	58	0	16	57	0
H5	5	0	0	9	0	0	0	0	0	3	0	0
H6	472	68	169	458	55	193	174	58	91	448	69	171
H7	37	13	6	44	15	4	31	14	7	30	15	10
H8	1	0	0	0	0	0	0	0	0	0	0	0
H9	1	1	2	5	0	2	3	0	3	4	0	2
H10	8	30	16	9	28	17	3	30	7	15	27	21
H11	8	0	1	6	0	3	8	0	2	9	0	2

The first step of our experimental procedure is to generate all possible FOMs, and our purpose is to compare the number of generated FOMs with the number of generated HOMs. From the set of FOMs, HOMs were generated and evaluated by applying multi-objective optimization algorithms, guided by objectives and fitness functions. According to the results which are presented in Table 7 and Table 8, the number of generated HOMs was reduced at least 70% compared with the produced first order mutants. Reduction number of generated mutants leads to reduction the cost of execution of mutation testing.

The results presented in Figure 2 indicate that the majority of generated HOMs are H1 (equivalent mutants) and H6 (Non-Quality, Reasonable and With Old TCs.). A high number of H6 shows that there are many generated HOMs which are more difficult to be killed than FOMs and only be killed by TCs belonging to the union of sets of TCs that can kill their constituent FOMs, except the TCs that can kill simultaneously all their constituents (The mutants of H6 only be killed by the TCs belonging to set C, see Figure 1 in Third section). The mean proportion of H7 in total of generated HOMs is similar to H3, H4 and H10, around from 4% to 5.7%, whilst the numbers of H2, H5, H8, H9 and H11 are very small. Even only one H8 was generated in our experiment.

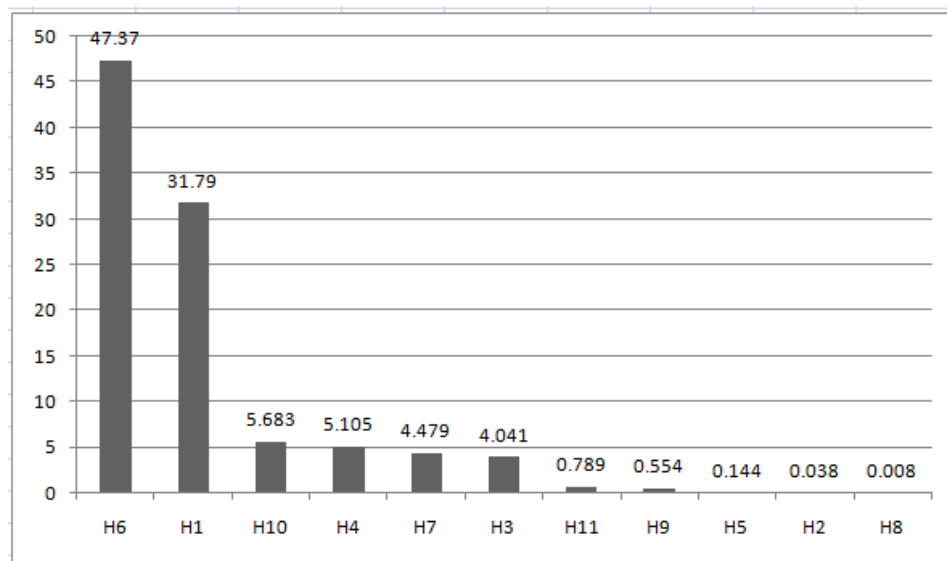


Figure 2. The mean proportion of 11 categories of HOMs in total of HOMs(%)

Answer to RQ2

RQ2 is aimed to evaluate percentage of Non-quality, Un-reasonable HOMs (H2 and H3). These HOMs are “not good” because they are the HOMs which are easier to kill than their constituent FOMs and have no any test case which can kill them and simultaneously all their constituent FOMs. Our results indicate that the percentage of these HOMs is not large. The mean percentages of Non-quality, Un-reasonable HOMs corresponding to the Barbecue, BeanBin and JWBF projects are around 8.97%, 1.14%, 2.13% respectively (see Figure 3).

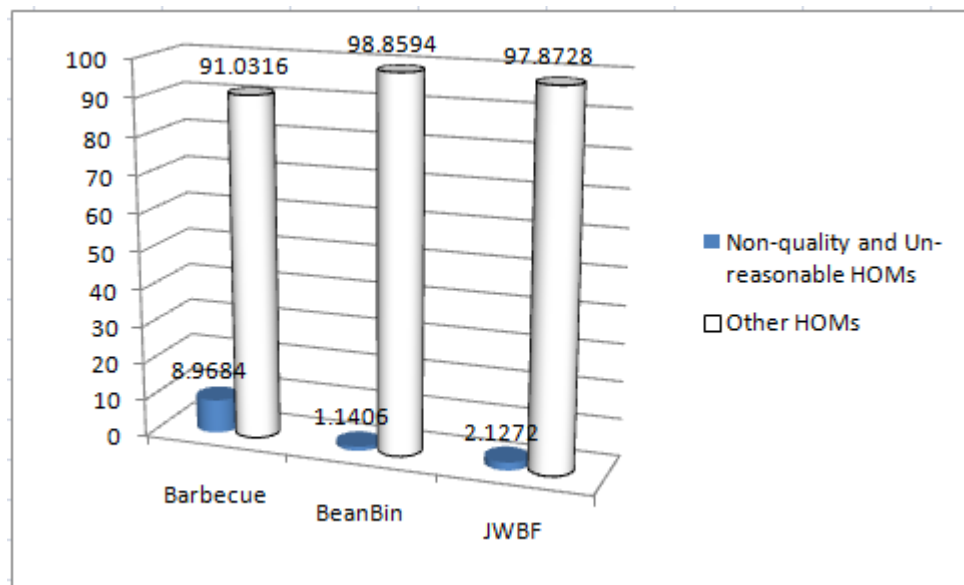


Figure 3. The mean proportion of Non-quality, Un-reasonable HOMs (H2 and H3) and other HOMs (%) to all generated HOMs

Answer to RQ3

The ratio of reasonable HOMs (H4-H9) to total generated HOMs is fairly high, over 56% of total generated HOMs (see Table 9). This indicates that we can find the mutants that are harder to kill and more realistic (reflecting real, complex faults) than FOMs by applying multi-objectives optimization algorithm.

Table 9. The ratios of Reasonable HOMs and Equivalent HOMs to generated HOMs

Algorithm	% Reasonable HOMs	% Equivalent HOMs
eMOEA	57.91	32.89
NSGA-II	56.80	32.69
eNSGA-II	58.49	31.68
NSGA-III	57.16	32.13

Figure 4 shows the percentage of high quality and reasonable HOMs (H7) compared with total generated HOMs and the number of reasonable HOMs. The algorithm eNSGA-II is the best for searching for high quality and reasonable HOMs. Approximately 11% of reasonable HOMs, which were found by eNSGA-II algorithm, are classified as high quality and reasonable HOMs. According to the results of four applied algorithms, the mean ratio of high quality and reasonable HOMs to all found reasonable HOMs is about 8.74%. This number is high because the proportion of all reasonable HOMs to all generated HOMs is a large number, over 56% (see Table 9). Hence, we believe that high quality and reasonable HOMs are not too hard to find and therefore HOMs which are constructed by applying multi-

objective optimization algorithm, especially high quality and reasonable HOMs, can be used to replace the set of their constituent FOMs without loss of test effectiveness.

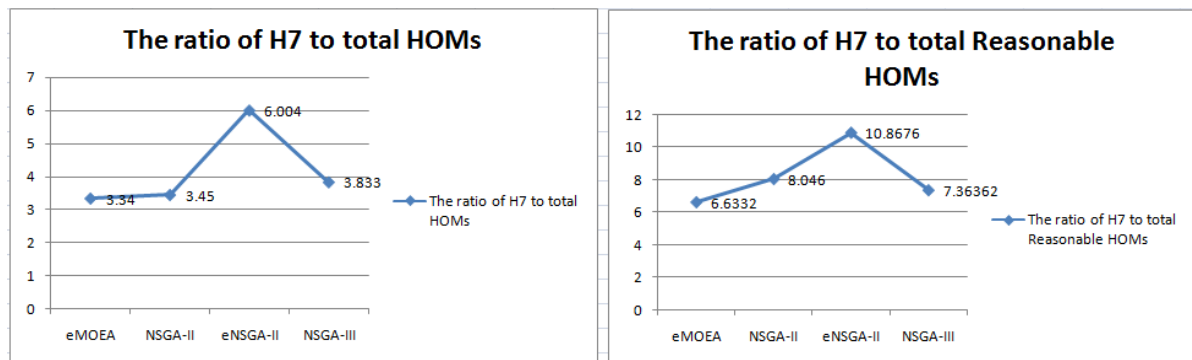


Figure 4. The ratios of H7 to total HOMs and Reasonable HOMs (%)

DISCUSSION

Our works based on the idea of application of multi-objective higher order mutation testing of Harman et al. (Jia and Harman 2008, 2009, Harman et al. 2010, Langdon et al. 2009, 2010). Although this is a promising approach, to the best of our knowledge, still there is little research in the field of applying multi-objective optimization algorithms in the field of higher order mutation testing. There are three papers, which were published by Harman et al. 2010, Langdon et al. 2009 and 2010, focused on multi-objective higher order mutation testing with genetic programming. However, they used only one multi-objective optimization algorithm, NSGAII, to search for higher order mutants which are both hard to kill and realistic. According to their results, the number of equivalent mutants falls rapidly but the number of mutants grows exponentially in order with the number of changes made.

We have proposed not only a new classification of HOMs based on the combination of a set of test cases (TCs), which kills HOM and sets of TCs which kill constituent FOMs but also new objectives and fitness functions, which are applied to multi-objective optimization algorithms for constructing HOMs. Our HOM classification can cover all of the available cases of generated HOMs, some of which were not mentioned in the classification of HOMs of Jia and Harman (2009) and Harman et al. (2010). In our proposed objective functions, we focus on constructing the HOMs, which are killed by as small sets of TCs as possible. It leads to reduce the number of generated HOMs. From that, we have investigated applying our proposing to multi-objective optimization algorithms for finding high quality and reasonable (strongly subsuming and coupled^(*)) HOMs. Our main goal, similar to other authors, is to carry out investigation and to propose the approach, which can be used to construct the difficulties to kill and more realistic HOMs, especially high quality and reasonable HOMs, as well as reduce the number of generated higher order mutants in particular. Hence, it could be used to improve the mutation testing effectiveness in general.

We have used a personal computer with Intel® Core™ i5-2410M CPU@2.30GHz and 4GB RAM to run our experiment. The mean time for running mutation testing for Barbecue, BeanBin and JWBF projects (for each project, *This is an Accepted Manuscript of an article published on 09 Feb 2016 by Taylor & Francis Group in Cybernetic and Systems, Vol. 47, Iss. 1-2 (2016), p. 48-68, available online: <http://dx.doi.org/10.1080/01969722.2016.1128763>*

we ran 5 algorithms) is 112.8 seconds, 32 seconds and 33 seconds. The time includes generation, evaluation, compilation and execution. The results of our experiment have indicated that the number of total generated HOMs was reduced at least 70% compared with FOMs. It means that our approach can be used to overcome the first limitation of the traditional mutation testing. This limitation is a large number of mutants and this also leads to have a very high execution cost. We have also indicated that it is not too hard to find high quality and reasonable HOMs which are both harder to kill and more realistic than FOMs. They can be used to replace the set of their constituent FOMs without loss of test effectiveness. In addition to this, we have reported a low number of “not good” HOMs - the HOMs which are easier to kill than their constituent FOMs and the set of test cases which kill these HOMs cannot kill simultaneously their constituent FOMs.

However the number of equivalent HOMs in our experiment is still large. In this case, we need further more researches to evaluate whether the HOMs are really equivalent mutants or not. There may exist some HOMs that are not killed by the given set of TCs but can be killed by some other new TCs (Omar et al. 2013). In this case, we have to create new TCs to improve the fault detection effectiveness of the existing set of TCs.

We believe that we have exposed one of the promising approaches in the area of higher order mutation testing (the genuine idea by Jia and Harman (2009) and Harman et al. (2010)). Classifying and identifying HOMs by values of our objective and fitness functions which we apply in multi-objective optimization algorithms for finding valuable high quality and reasonable HOMs are useful in mutation testing.

THREATS TO VALIDITY

There is always a set of threats to the validity of the results. We identified some threats to the validity following the guidelines by Wohlin et al. (2012) and Madeyski (2010a).

We identify some threats to the validity, which we will discuss in this section. A threat to external validity is that the selected subject programs may not be representative of Java programs in general, and therefore, the results of the study may not be generalizable to all Java programs. However, the selected SUTs are different in functionality. Furthermore, their size is comparable to or bigger than many of SUTs used in previous empirical studies by the mutation testing research community.

Another threat to external validity stems from the mutation operators that were used to generate mutants. Using different tools that implement different mutation operators to generate mutants can lead to different results including different distribution of HOMs. However, the set of mutation operators supported by Judy (Madeyski et al. 2014) is rather wide in comparison to sets analyzed in other empirical studies investigating higher order mutation. Albeit, one may want to extend the set even further.

A threat to internal validity stems from the quality and the coverage of test cases. There is a possibility of getting different results if test suites of higher and lower quality or coverage were used.

A threat to construct validity stems from the correctness of the implementation of Judy and the manual identification of equivalent mutants, which is a non-trivial and time-consuming activity (Madeyski et al. 2014).

CONCLUSIONS AND FUTURE WORK

Based on our classification of HOMs as well as our objective and the fitness functions, we apply four different multi-objective optimization algorithms to search for high quality and reasonable HOMs (strongly subsuming and coupled HOMs^(*)) and ten other types of HOMs. In our investigation, the main goal is to reduce the number of generated mutants and to find the high quality and reasonable HOMs. The results of our experiment indicate that our approach is able to reduce the generated HOMs compared with FOMs. Besides, the results also indicate that our approach could be useful in searching for available high quality and reasonable HOMs because all of four proposed algorithms, guided by our objective functions and fitness functions, found high quality and reasonable HOMs.

The number of equivalent HOMs is still large. In this case, equivalent HOMs cannot be killed by any test cases of the given set of test cases T (see second Section). It means that they could be killed by some test cases that do not belong to T. So, in the future, further research is needed for improving our approach to assess whether HOMs are equivalent or not. In addition, we will apply the other search-based algorithms based on our proposed objective and fitness functions for comparing the effectiveness of different algorithms for high quality and reasonable HOMs.

REFERENCES

- Adekanmbi, O.A., Olugbara, O.O., Adeyemo, J. 2014. A Comparative Study of State-of-the-Art Evolutionary Multi-objective Algorithms for Optimal Crop-mix planning. International Journal of Agricultural Science and Technology (IJAST) Volume 2 Issue 1. DOI: 10.14355/ijast.2014.0301.02
- Deb, K., Jain, H. 2014. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints, IEEE Transactions on evolutionary computation, Vol. 18, No. 4.
- Deb, K., Mohan, M., Mishra, S. 2003. A Fast Multi-objective Evolutionary Algorithm for Finding Well-Spread Pareto-Optimal Solutions. KenGAL, Report No. 2003002. Indian Institute of Technology, Kanpur, India.
- Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE transactions on evolutionary computation, vol. 6, No. 2.
- DeMillo, R.A., Lipton R.J., Sayward, F.G. 1978. Hints on test data selection: help for the practicing programmer. IEEE Computer 11 (4), 34–41.
- Hamlet, R.G. 1977. Testing programs with the aid of a compiler. IEEE Transactions on Soft. Eng. SE-3 (4), 279–290.

- Harman, M., Jia, Y., Langdon, W. B. 2010. A Manifesto for Higher Order Mutation Testing. In: Third International Conf. on Software Testing, Verification, and Validation Workshops.
- Jia, Y., Harman, M. 2008. Constructing Subtle Faults Using Higher Order Mutation Testing. In: Proc. Eighth Int'l Working Conf. Source Code Analysis and Manipulation.
- Jia, Y., Harman, M. 2009. Higher order mutation testing. *Information and Software Technology* 51, 1379–1393.
- Kintis, M., Papadakis, M., Malevris, N. 2010. Evaluating mutation testing alternatives: A collateral experiment. In: Proc. 17th Asia Pacific Soft. Eng. Conf. (APSEC)
- Kollat, J.B., Reed, P.M. 2005. The Value of Online Adaptive Search: A Performance Comparison of NSGAII, ϵ -NSGAII and ϵ -MOEA, Carlos A. CoelloCoello Arturo Hernández Aguirre EckartZitzler (Eds.), *Evolutionary Multi-Criterion Optimization*, Third International Conference, EMO 2005 Guanajuato, Mexico.
- Langdon, W.B., Harman, M., Jia, Y. 2009. Multi Objective Higher Order Mutation Testing with Genetic Programming. In: Proc. Fourth Testing: Academic and Industrial Conf. Practice and Research.
- Langdon, W.B., Harman, M., Jia, Y. 2010. Efficient multi-objective higher order mutation testing with genetic programming. *The Journal of Systems and Software* 83.
- Madeyski, L. 2007. On the effects of pair programming on thoroughness and fault-finding effectiveness of unit tests, *Lecture Notes in Computer Science*, vol. 4589, pp. 207–221. DOI: 10.1007/978-3-540-73460-4_20
- Madeyski, L. 2008. The impact of pair programming on thoroughness and fault detection effectiveness of unit tests suites, Wiley, *Software Process: Improvement and Practice*, vol. 13, no. 3, pp. 281–295. DOI: 10.1002/spip.382
- Madeyski, L. 2010a. *Test-Driven Development: An Empirical Evaluation of Agile Practice*, Springer. DOI: 10.1007/978-3-642-04288-1.
- Madeyski, L. 2010b. The impact of test-first programming on branch coverage and mutation score indicator of unit tests: An experiment. *Information and Software Technology*, vol. 52, no. 2, pp. 169–184. DOI: 10.1016/j.infsof.2009.08.007.
- Madeyski, L., Orzeszyna, W., Torkar, R., Józala, M. 2014. Overcoming the Equivalent Mutant Problem: A Systematic Literature Review and a Comparative Experiment of Second Order Mutation. *IEEE Transactions on Software Engineering*, 40 (1), pp. 23-42. DOI: 10.1109/TSE.2013.44.
- Madeyski, L., Radyk, N. 2010. Judy - a mutation testing tool for Java. *IET Software* 4(1): 32-42 (2010). DOI: 10.1049/iet-sen.2008.0038.
- Nguyen, Q.V., Madeyski, L. 2014. Problems of Mutation Testing and Higher Order Mutation Testing. *Advanced Computational Methods for Knowledge Engineering, Advances in Intelligent Systems and Computing* 282, Springer. DOI: 10.1007/978-3-319-06569-4_12.

- Nguyen, Q.V., Madeyski, L. 2015. Searching for Strongly Subsuming Higher Order Mutants by Applying Multi-objective Optimization Algorithm, in Advanced Computational Methods for Knowledge Engineering (H. A. Le Thi, N. T. Nguyen, and T. V. Do, eds.), vol. 358 of Advances in Intelligent Systems and Computing, pp. 391–402, Springer International Publishing. DOI:10.1007/978-3-319-17996-4_35.
- Omar, E., Ghosh, S., Whitley, D. 2013. Constructing subtle higher order mutants for Java and AspectJ programs, Software Reliability Engineering (ISSRE), IEEE 24th International Symposium.
- Papadakis, M., Malevris, N. 2010. An empirical evaluation of the first and second order mutation testing strategies. In: Proceedings of Third Inter. Conf. on Software Testing, Verification, and Validation Workshops, ser. ICSTW'10, IEEE Computer Society, pp. 90–99.
- Papadakis, M., Yue, J., Harman M., and Traon, Y.L. 2015. Trivial Compiler Equivalence: A Large Scale Empirical Study of a Simple, Fast and Effective Equivalent Mutant Detection Technique, in 37th (ICSE).
- Polo, M., Piattini, M., Garcia-Rodriguez, I. 2008. Decreasing the Cost of Mutation Testing with Second-Order Mutants. Software Testing, Verification, and Reliability, vol. 19, no. 2, pp. 111-131.
- Purushothaman, R., Perry, D.E. 2005. Toward Understanding the Rhetoric of small source code changes. IEEE Transactions on Software Engineering 31(6).
- Wohlin, C., Runeson, P., Host, M., Ohlsson, M. C., Regnell, B., Wesslen, A. 2012. Experimentation in Software Engineering: An Introduction. Berlin Heidelberg, Germany: Springer, 2 ed.