On the relationship between the order of mutation testing and the properties of generated higher order mutants

Quang Vu Nguyen and Lech Madeyski

Faculty of Computer Science and Management, Wroclaw University of Technology, Poland

Abstract. The goal of higher order mutation testing is to improve mutation testing effectiveness in particular and test effectiveness in general. There are different approaches which have been proposed in the area of second order mutation testing and higher order mutation testing with mutants order ranging from 2 to 70. Unfortunately, the empirical evidence on the relationship between the order of mutation testing and the desired properties of generated mutants is scarce except the conviction that the number of generated mutants could grow exponentially with the order of mutation testing. In this paper, we present the study of finding the relationships between the order of mutation testing and the properties of mutants in terms of number of generated high quality and reasonable mutants as well as generated live mutants. Our approach includes higher order mutants classification, objective functions and fitness functions to classify and identify generated higher order mutants. We use four multi-objective optimization algorithms for constructing higher order mutants. Obtained empirical results indicate that 5 is a relevant highest order in higher order mutation testing.

Keywords: Mutation Testing, Higher Order Mutation, Higher Order Mutants, Multi-objective optimization algorithm.

1 Introduction

Mutation testing has been considered as one of the most effective techniques for evaluating the quality of given sets of test data which are used in software testing. The technique is applied to assess the quality of given sets of test cases (TCs) based on their ability of detecting the differences between program under test (PUT) and its mutants [1,2]. The mutants are different PUT versions, which are produced by syntactically altering the source code of the PUT. The syntactic changes are called mutation operators. After executing the given set of test cases on the original program (PUT) and each of its mutants, mutation testing evaluates the quality of test cases by mutation score (MS) or mutation score indicator (MSI). MS is defined as the ratio of killed mutants to the differences of all generated mutants and equivalent mutants [1,2]. While MSI is the ratio of killed mutants to all generated mutants [9,10,11,12].

Many approaches (e.g., selective mutation, sampling mutation and weak mutation) have been proposed to overcome limitations of mutation testing [6,14]. Higher order mutation testing, an idea of Jia and Harman in 2009 [5,3], is one of the most promising solutions. Instead of using only one simple change as the traditional mutation testing [1,2], higher order mutation testing uses more complex changes to generate mutants by applying two or more mutation operators. An n-order mutant is created by n mutation operators, for example, it can be generated by combining n first order mutants. Hence, higher order mutants reflect more realistic complex faults and can be harder to kill than first order mutants [5,3,8,4]. Strongly subsuming higher order mutants (HOMs) [5,3] can be used to replace all of n constituent first order mutants (FOMs). This is not only without loss of test effectiveness but potentially can reduce the cost of mutation testing execution by reducing the number of generated mutants.

Equivalent mutant is the one that has the same semantic meaning as the original program and thus cannot be detected any test suite [1,2]. Higher order mutation testing can also be helpful to overcome equivalent mutants problem (EMP) [12] which is a serious, long-standing problem of mutation testing.

In this paper, our research focuses on finding the “relevant highest order” of higher order mutants based on the relationships between the order of mutation testing and the properties of mutants. We apply multi-objective optimization algorithms to search for valuable HOMs and investigate the relationships between the order of mutation testing and the properties of mutants in term of ability for constructing high quality and reasonable HOMs, as well as generating live HOMs. High quality and reasonable HOM, one of 11 HOM types that were classified by us [15,16], is a HOM which is harder to kill than any constituent FOMs and is only killed by the subset of the intersection of set of test cases that kill each constituent FOM. This definition is the same as the definition of Strongly subsuming and coupled HOM in the classification by Harman et al. [5,3] which we extended. Live mutants are the mutants which cannot be killed by the given test suite but could be killed by new quality TCs [17]. In this case, we have to create new TCs to improve the fault detection effectiveness of the existing set of TCs.

The rest of the paper is organized as follows. Section 2 is the overview of the proposed approaches in higher order mutation testing. Section 3 presents the experiment goals, the multi-objective optimization algorithms and implementation details. Section 4 includes the results to answer the posed research questions. The last section includes the conclusions and further work.

2 Related work

Second order mutation testing is a specific case of higher order mutation testing. According to results of works on second order mutation testing, not only at least 50% of mutants were reduced [19,18,12] without loss of effectiveness of testing, 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 [19] or the mean reduction of equivalent HOMs is about 50% compared with FOMs [12]) and generated second order mutants can be harder to kill than first order mutants [12][19][18][7].

In 2009, Jia and Harman [5][3] defined a new paradigm for mutation testing—higher order mutation testing—and the rules to classify HOMs. They then used search-based optimization algorithms to find and evaluate the proportion of subsuming, as well as strongly subsuming HOMs to all generated HOMs. Their experiments showed approximately 15% of all found Subsuming HOMs are strongly subsuming HOMs and they also indicated that finding such HOMs may not be too difficult. The highest order of their experiment is 9 and they summarized that "the highest order mutants may find application in attempts to reduce mutation effort because they subsume the largest number of FOMs" [3].

Langdon et al. [8] suggested inserting "semantically close" faults instead of inserting "syntactically close" faults to the original program under test in order to produce better mutants. Their opinions are based on the claim of Purushothaman and Perry [20], who indicated that a modification made to fix one real fault needs several source code changes. From analysis of the relationship between syntax and semantics, Langdon et al. [8] introduced two objectives, small semantic distance and minimum syntactic changes, which were applied using NSGA-II multi-objective optimization algorithm. Their goal is to find higher order mutants that represent more realistic complex faults and are harder to kill. The highest order of their experiment is 70 and the number of mutants grows exponentially with order.

Omar et al. [17] presented the approach using search-based algorithms for finding subtle HOMs with a new objective function to identify subtle HOMs. They defined subtle HOMs as HOMs that are not killed by a given set of test cases but can be killed by other new test cases. They set up different maximum orders for each algorithm. For example, 25 is the maximum HOM degree for Random Search Algorithm and 15 is the maximum HOM degree for Genetic Algorithm. Their results indicate that the ability in finding subtle HOMs of lower degrees or higher degrees belongs to different algorithms [17].

In our previous work [15,16], with 15 being the highest considered degree of mutants, we used multi-objective optimization algorithms for finding valuable high quality and reasonable HOMs (strongly subsuming and coupled HOMs) based not only on a new classification of HOMs but also new objective and fitness functions. The results indicated that our approach can be useful in searching for available high quality and reasonable HOMs, and among them, eNSGA-II is one of the best algorithms. In this paper we use the classification of HOMs, as well as objective and fitness functions proposed by us [15,16]. There are eleven categories of HOMs (see Table 1) and they are identified on a basis of the values of 4 fitness functions [15][16].

Table 1: Eleven categories of HOMs (see [16] for details)

Name	HOM is

H1	Live (potentially 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	Non-Quality, Reasonable and With Old TCs
H7	**High quality and Reasonable**
H8	Quality, Reasonable and With Mixed TCs
H9	Quality, Reasonable and With Old TCs
H10	Quality, Un-Reasonable and With Old TCs
H11	Quality, Un-Reasonable and With Mixed TCs

H1 category describes live (and potentially equivalent) mutants, which cannot be killed by the given set of test cases.

3 Experiment goals and set up

3.1 Goals

The study will answer the research questions posed as follows:

RQ1. What are the ratios of number of HOMs in the identified mutant categories (H1-H11) to all generated HOMs for different orders?

By means of this question, we want to obtain the number of generated HOMs in the identified mutant categories. A number of generated HOMs will be collected and classified according to the kind of HOMs and according to the order of HOMs.

RQ2. What are the ratios of high quality and reasonable HOMs (H7) to all generated HOMs for different orders?

This question is, in fact, a part of the previous research question focused on a kind of mutants being of special interest. The aim is to obtain the frequency of generating high quality and reasonable HOMs (H7). Such mutants not only reflect harder to kill, realistic, complex faults but also could be used to replace all of its constituent FOMs.

RQ3. What are the ratios of "live (potentially equivalent) mutants" (H1) to all generated HOMs for different orders?

Answering this question may shed some light while trying to find the relevant highest order of mutation testing.

3.2 Experimental units and material

In this study we use the same mutation testing tool for Java called Judy, including also multi-objective optimization algorithms for searching HOMs, and three different projects under test for this study as in our previous works [15,16].

Judy[1] [12,13] mutation testing tool for Java not only provide the large set of mutation operators but also has build-in support for HOMs generation, higher order mutation testing execution and mutation analysis.

Four multi-objective optimization algorithms (NSGA-II, eNSGA-II, NSGA-III and eMOEA) are implemented to produce and evaluate HOMs based on our objective and fitness functions [16].

In our empirical study, we use three projects under test (PUT) [16], which are real-world software projects. Table 2 shows lines of code (LOC), number of classes (NOC) and number of given test cases (NOT) of the three selected open source projects.

Table 2: Software projects under test

Project Under Test (PUT)	LOC	NOC	NOT
BeanBin[2]	5925	72	68
Barbecue[3]	23996	57	190
JWBF (Java Wiki Bot Framework)[4]	13572	51	305

3.3 Approach

We set out the experimental procedure as follows (for each software, we run each algorithm 3 times, after then we calculate the average numbers to evaluate):

```
for each software under test do
    for each algorithm do
        loop 3 times
            .generate all possible FOMs by applying
            the set of Judy mutation operators
            .set objective and fitness functions
            for each multi-objective optimization algorithm do
                .set populationSize =100
                .set maxMutationOrder =15
                .from set of FOMs, generate and evaluate HOMs,
                guided by objectives and fitness functions
                .calculate the numbers to answer RQs
            end for
        end loop
        .calculate the mean values
    end for
end for
```

[1] http://www.mutationtesting.org/
[2] http://beanbin.sourceforge.net
[3] http://barbecue.sourceforge.net
[4] http://jwbf.sourceforge.net

4 Results and analysis

The maximum mutation order in our experiment is 15. Differences in the source code of the three SUTs resulted in some variations in mutation operators finally used to generate HOMs [15][16].

Answer to RQ1

To answer this question, we calculate the ratios of number of HOMs in each of the identified mutant categories (H1-H11) to all generated HOMs (see Table 3), as well as the ratios of number of HOMs of a particular order (2-15) to all generated HOMs (see Table 4).

Table 3: The ratios of number of HOMs in the identified mutant categories to all generated HOMs [%]

HOMs	eMOEA	NSGAII	eNSGAII	NSGAIII
H1	35.58	34.16	37.61	34.26
H2	0.06	0.06	0	0.14
H3	5.11	6.72	4.18	6.28
H4	4.32	3.66	6.89	4.81
H5	0.33	0.59	0	0.2
H6	46.45	46.11	38.46	45.41
H7	3.68	4.12	6.18	3.62
H8	0.06	0	0	0
H9	0.26	0.45	0.71	0.4
H10	3.54	3.53	4.75	4.16
H11	0.59	0.59	1.18	0.73

Table 4: The ratios of number of HOMs of a particular order (2-15) to all generated HOMs [%]

Order	eMOEA	NSGAII	eNSGAII	NSGAIII
2	11.65	12.57	9.34	12.18
3	15.67	14.70	14.12	14.82
4	14.69	16.11	14.01	14.94
5	13.57	15.08	14.73	12.56
6	11.07	8.93	7.44	9.39
7	7.81	6.93	7.08	8.53
8	5.96	5.57	3.34	5.22
9	4.13	4.47	5.28	4.31
10	3.60	4.08	3.13	3.51
11	2.10	2.52	5.14	3.37
12	3.01	3.55	5.03	2.84
13	2.81	1.49	4.67	3.31
14	1.63	2.19	3.47	2.38
15	2.30	1.81	3.22	2.64

The ratios of number of HOMs in the identified mutant categories to all generated HOMs are similar for four algorithms, see also Figure 1.

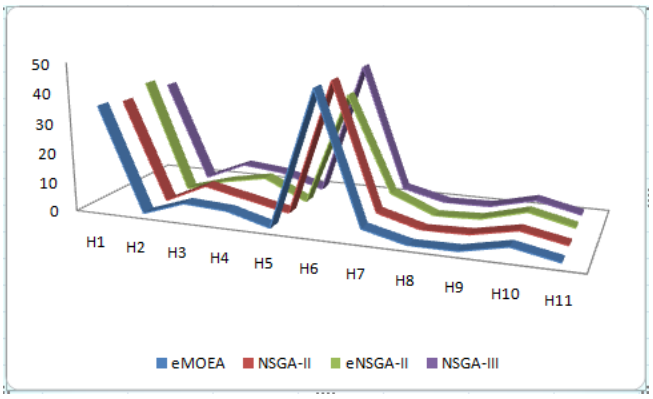

Fig. 1. The ratios of number of (H1-H11) to all generated HOMs [%]

The mean ratio of H1 mutants to all HOMs is around 35% and the ratio of high quality and reasonable HOMs to all HOM is from 3.68% to 6.42%. H1 mutants in this case are live mutants, which cannot be killed by the given test suite but could be killed by some other new quality TCs [17]. We need further investigation to evaluate whether the HOMs are really equivalent mutants or not. It includes creating the new quality TCs to detect the difference between PUT and its non-equivalent mutants which belong to the set of live mutants.

A high number of H6 (see Table 3 and Figure 1) 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 ratio of total of reasonable HOMs (H4-H9) to all of generated HOMs is fairly high, over 55% of total generated HOMs. 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. Approximately 9% of reasonable HOMs (H4-H9) are classified as high quality and reasonable HOMs (H7). This number is high because the ratio of all reasonable HOMs to all generated HOMs is quite a large.

Table 4 describes the ratios of generated HOMs of a particular order to all of generated HOMs. The results indicated that generally for lower orders the number of generated HOMs is larger than for higher orders, for all of our four search-based algorithms.

Answer to RQ2

High quality and reasonable HOMs (H7) are the HOMs which are more realistic complex faults and harder to kill than any FOMs [16,15]. In addition, using them to replace all of its constituent FOMs leads to reducing testing costs without loss of test effectiveness. Obtained empirical results show that we can find high quality and reasonable HOMs from the 2nd-order to the 5th. For the 6th-order, as well as for higher orders, generated high quality and reasonable HOMs are rare. There is lack of high quality and reasonable HOMs in many cases (see Table 5). As a result, we may conclude that higher order mutation up to the 5-th order can be rewarding wrt. searching for high quality and reasonable HOMs (H7) by applying multi-objective optimization algorithms.

Table 5: The mean ratios of H7 to all generated HOMs per order [%]

Order	2	3	4	5	6	7	8	9	10	11	12	13	14	15
eMOEA	11.80	6.27	5.76	1.88	0.53	0.00	0.99	0.00	1.64	0.00	0.00	0.00	0.00	0.00
NSGAII	14.37	7.53	3.98	2.19	0.65	0.84	0.00	1.30	0.00	0.00	0.00	0.00	0.00	0.00
eNSGAII	20.38	11.96	8.46	4.88	0.00	0.00	0.00	0.00	0.00	0.00	2.14	0.00	0.00	0.00
NSGAIII	13.05	6.69	3.98	1.11	0.00	0.70	0.00	1.38	0.00	0.00	0.00	0.00	0.00	0.00

Answer to RQ3

Table 6 shows the mean ratios of live (and potentially equivalent) mutants (H1) to all produced HOMs according to orders. The number of H1 mutants is quite large–22 to 55% of the generated HOMs. Live mutants include non-equivalent mutants and equivalent mutants. Non-equivalent mutants can be killed by some new quality TCs. Equivalent mutants are really same-semantic-meaning versions of the original program under test and cannot be killed by any test suite. In this case, we need a further investigation to evaluate whether live mutants are equivalent or not (a thorough review of the possible approaches and their classification is presented by Madeyski et al. [12]). This leads to creating new high quality TCs to improve the fault detection effectiveness of the existing set of test cases.

Table 6: The mean ratios of H1 to all generated HOMs per order [%]

Order	2	3	4	5	6	7	8	9	10	11	12	13	14	15
eMOEA	59.53	41.41	33.07	22.75	25.40	35.26	32.01	36.67	27.32	37.38	30.72	37.06	36.14	40.17
NSGAII	50.54	43.59	32.53	26.13	28.91	29.97	38.33	27.39	33.33	30.77	27.32	35.06	26.55	32.26
eNSGAII	58.85	50.13	26.41	22.68	22.71	30.46	35.48	40.82	42.53	53.85	42.86	46.15	41.24	36.67
NSGAIII	48.45	42.24	28.82	29.54	23.89	33.26	26.62	27.65	41.24	37.06	48.95	22.16	35.83	30.08

FINDING: *5 is a relevant highest order in higher order mutation testing as the ratio of high quality and reasonable HOMs (H7) to total number of HOMs (generated using multi-objective optimization algorithms) is high for orders between 2 and 5. This ratio is low, close to zero, for orders higher than 5, while the ratio of live (and potentially equivalent) mutants to total number of HOMs is large for every order* (see Figure 2).

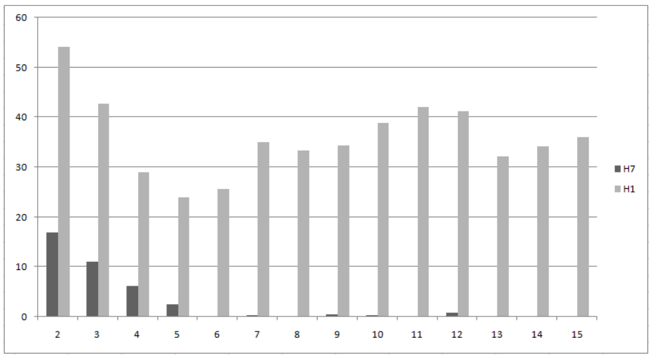

Fig. 2. The ratios of H1 and H7 to all generated HOMs per order [%]

5 Conclusions and future work

In this paper, we have investigated the relationships between the order of mutation testing and the properties of generated higher order mutants. We evaluated the results on a basis of generated different kinds of HOMs (H1-H11), especially the number of generated high-quality-reasonable HOMs (H7) and the number of generated live HOMs (H1). The empirical results indicated that 5 can be a relevant highest order in higher order mutation testing.

Using only three selected projects under test (PUTs) may not be a representative sample of all Java programs in general and therefore, the results of the study may not be generalizable to all Java programs as well as other programming language. Hence further research is recommended. Nevertheless, we believe that this study is a step towards unveiling the relationship between the order of mutation testing and the properties of generated mutants represented by our classification of higher order mutants.

References

1. DeMillo, R.A., Lipton, R.J., Sayward, F.G.: Hints on Test Data Selection: Help for the Practicing Programmer. IEEE Computer 11(4), 34–41 (1978)
2. Hamlet, R.G.: Testing Programs with the Aid of a Compiler. IEEE Transactions on Software Engineering 3(4), 279–290 (1977)
3. Harman, M., Jia, Y., Langdon, W.B.: A Manifesto for Higher Order Mutation Testing. In: Proceedings of the 2010 Third International Conference on Software Testing, Verification, and Validation Workshops. pp. 80–89. ICSTW '10, IEEE Computer Society, Washington, DC, USA (2010)
4. Jia, Y., Harman, M.: Constructing subtle faults using higher order mutation testing. In: Source Code Analysis and Manipulation. pp. 249–258 (2008)
5. Jia, Y., Harman, M.: Higher order mutation testing. Information and Software Technology 51(10), 1379–1393 (Oct 2009)
6. Jia, Y., Harman, M.: An Analysis and Survey of the Development of Mutation Testing. IEEE Transactions on Software Engineering 37(5), 649–678 (Sep 2011)

7. Kintis, M., Papadakis, M., Malevris, N.: Evaluating mutation testing alternatives: A collateral experiment. In: Software Engineering Conference (APSEC), 2010 17th Asia Pacific. pp. 300–309 (Nov 2010)
8. Langdon, W.B., Harman, M., Jia, Y.: Efficient multi-objective higher order mutation testing with genetic programming. Journal of Systems and Software 83(12), 2416–2430 (Dec 2010)
9. Madeyski, L.: On the effects of pair programming on thoroughness and fault-finding effectiveness of unit tests. In: Münch, J., Abrahamsson, P. (eds.) Product-Focused Software Process Improvement, Lecture Notes in Computer Science, vol. 4589, pp. 207–221. Springer Berlin Heidelberg (2007), doi:10.1007/978-3-540-73460-4_20
10. Madeyski, L.: Impact of pair programming on thoroughness and fault detection effectiveness of unit test suites. Software Process: Improvement and Practice 13(3), 281–295 (2008), doi:10.1002/spip.382, draft: `http://madeyski.e-informatyka.pl/download/Madeyski08.pdf`
11. Madeyski, L.: The impact of test-first programming on branch coverage and mutation score indicator of unit tests: An experiment. Information and Software Technology 52(2), 169–184 (2010), doi:10.1016/j.infsof.2009.08.007
12. Madeyski, L., Orzeszyna, W., Torkar, R., Józala, M.: Overcoming the Equivalent Mutant Problem: A Systematic Literature Review and a Comparative Experiment of Second Order Mutation. IEEE Transactions on Software Engineering 40(1), 23–42 (1 2014), doi:10.1109/TSE.2013.44
13. Madeyski, L., Radyk, N.: Judy – A Mutation Testing Tool for Java. IET Software 4(1), 32–42 (2010), doi:10.1049/iet-sen.2008.0038
14. Nguyen, Q.V., Madeyski, L.: Problems of mutation testing and higher order mutation testing. In: Do, T., Thi, H.A.L., Nguyen, N.T. (eds.) Advanced Computational Methods for Knowledge Engineering, Advances in Intelligent Systems and Computing, vol. 282, pp. 157–172. Springer International Publishing (2014), doi:10.1007/978-3-319-06569-4_12
15. Nguyen, Q.V., Madeyski, L.: Searching for Strongly Subsuming Higher Order Mutants by Applying Multi-objective Optimization Algorithm. In: Le Thi, H.A., Nguyen, N.T., Do, T.V. (eds.) Advanced Computational Methods for Knowledge Engineering, Advances in Intelligent Systems and Computing, vol. 358, pp. 391–402. Springer International Publishing (2015), doi:10.1007/978-3-319-17996-4_35
16. Nguyen, Q.V., Madeyski, L.: Empirical Evaluation of Multi-Objective Optimization Algorithms Searching For Higher Order Mutants. Cybernetics and Systems, An International Journal (2016), doi:10.1080/01969722.2016.1128763, (accepted) Draft: `http://madeyski.e-informatyka.pl/download/NguyenMadeyski16CS.pdf`
17. Omar, E., Ghosh, S., Whitley, D.: Constructing subtle higher order mutants for Java and AspectJ programs. In: Software Reliability Engineering (ISSRE), 2013 IEEE 24th International Symposium on. pp. 340–349 (Nov 2013)
18. Papadakis, M., Malevris, N.: An empirical evaluation of the first and second order mutation testing strategies. In: Software Testing, Verification, and Validation Workshops (ICSTW), 2010 Third International Conference on. pp. 90–99 (April 2010)
19. Polo, M., Piattini, M., García-Rodríguez, I.: Decreasing the cost of mutation testing with second-order mutants. Software Testing, Verification and Reliability 19(2), 111–131 (2009), doi:10.1002/stvr.392
20. Purushothaman, R., Perry, D.E.: Toward Understanding the Rhetoric of Small Source Code Changes. IEEE Transactions on Software Engineering 31(6), 511–526 (Jun 2005)