Metamorphic Testing: A Literature Review

Version 1.2

Sergio Segura, Gordon Fraser, Ana B. Sánchez and Antonio Ruiz-Cortés
sergiosegura@us.es

Applied Software Engineering Research Group
University of Seville, Spain
January 11, 2016

Technical Report ISA-16-TR-01

This report was prepared by the

Applied Software Engineering Research Group (ISA)
Department of computer languages and systems
Av/ Reina Mercedes S/N, 41012 Seville, Spain
http://www.isa.us.es/

Support: This work has been partially supported by the European Commission (FEDER) and Spanish Government under CICYT project TAPAS (TIN2012-32273) and the Andalusian Government projects THEOS (TIC-5906) and COPAS (P12-TIC-1867).

List of changes

Version	Date	Description
1.0	May 25, 2015	First release.
1.1	July 10, 2015	Eight new papers added to the review. New author added. Structure and writing updated.
1.2	January 11, 2016	Minor fixes throughout the paper. China and Hong Kong considered as two different countries. Search time span extended to November 2015. Seventeen new papers reviewed.

Metamorphic Testing: A Literature Review

Sergio Segura, Gordon Fraser, Ana B. Sanchez, and Antonio Ruiz-Cortés

Abstract—A test oracle determines whether a test execution reveals a fault, often by comparing the observed program output to the expected output. This is not always practical, for example when a program's input-output relation is complex and difficult to capture formally. *Metamorphic testing* provides an alternative, where correctness is not determined by checking an individual concrete output, but by applying a transformation to a test input and observing how the program output "morphs" into a different one as a result. Since the introduction of such *metamorphic relations* in 1998, many contributions on metamorphic testing have been made, and the technique has seen successful applications in a variety of domains, ranging from web services to computer graphics. This technical report provides a comprehensive literature review on metamorphic testing: It summarises the research results and application areas, and analyses common practice in empirical studies of metamorphic testing as well as the main open challenges.

Index Terms—Metamorphic testing, oracle problem, survey.

1 Introduction

Software testing is an essential but costly activity applied during software development to detect faults in programs. Testing consists of executing a program with test inputs, and to detect faults there needs to be some procedure by which testers can decide whether the output of the program is correct or not, a so–called *test oracle* [1]. Often, the test oracle consists of comparing an expected output value with the observed output, but this may not always be feasible. For example, consider programs that produce complex output, like complicated numerical simulations, or code generated by a compiler — predicting the correct output for a given input and then comparing it with the observed output may be non-trivial and error-prone. This problem is referred to as the *oracle problem* and it is recognised as one of the fundamental challenges of software testing [1], [2], [3], [4].

Metamorphic testing [5] is a technique conceived to alleviate the oracle problem. It is based on the idea that often it is simpler to reason about relations between outputs of a program, than it is to fully understand or formalise its input-output behaviour. The prototypical example is that of a program that computes the sine function: What is the exact value of $sin(12)$? Is an observed output of -0.5365 correct? A mathematical property of the sine function states that $\sin(x) = \sin(\pi - x)$, and we can use this to test whether $\sin(12) = \sin(\pi - 12)$ without knowing the concrete values of either sine calculation. This is an example of a *metamorphic relation*: an input transformation that can be used to generate new test cases from existing test data, and an output relation, that compares the outputs produced by a pair of test cases. Metamorphic testing does not only alleviate the oracle problem, but it can also be highly automated.

- *S. Segura, Ana. B. Sánchez and A. Ruiz-Cortés are with the Dept. of Computer Languages and Systems, Universidad de Sevilla, Spain. E-mail: sergiosegura@us.es*
- *G. Fraser is with the Dept. of Computer Science, the University of Sheffield, Sheffield, UK.*

The introduction of metamorphic testing can be traced back to a technical report by Chen et al. [5] published in 1998. However, the use of identity relations to check program outputs can be found in earlier articles on testing of numerical programs [6], [7] and fault tolerance [8]. Since its introduction, the literature on metamorphic testing has flourished with numerous techniques, applications and assessment studies that have not been fully reviewed until now. Although some papers present overviews of metamorphic testing, they are usually the result of the authors' own experience [9], [10], [11], [12], [13], review of selected articles [14], [15], [16] or surveys on related testing topics [3]. At the time of writing this technical report, the only known survey on metamorphic testing is written in Chinese and was published in 2009[1] [17]. As a result, publications on metamorphic testing remain scattered in the literature, and this hinders the analysis of the state of the art and the identification of new research directions.

In this technical report, we present an exhaustive literature review on metamorphic testing, covering 118 papers published between 1998 and 2015. To provide researchers and practitioners with an entry point, Section 2 contains an introduction to metamorphic testing. All papers were carefully reviewed and classified, and the review methodology followed in our literature review as well as a brief summary and analysis of the selected papers are detailed in Section 3. We summarise the state of the art by capturing the main advances on metamorphic testing in Section 4. Across all surveyed papers, we identified more than 12 different application areas, ranging from web services through simulation and modelling to computer graphics (Section 5). Of particular interest for researchers is a detailed analysis of experimental studies and evaluation metrics (Section 6). As a result of our literature review, a number of research challenges emerge, providing avenues for future research (Section 7); in particular, there are open questions on how to derive effective metamorphic relations, as well as how to

1. Note that 85 out of the 118 papers reviewed in our literature review were published in 2009 or later.

reduce the costs of testing with them.

2 METAMORPHIC TESTING

Identity relations are a well-known concept in testing, and have been used even before the introduction of metamorphic relations. For example, Blum et al. [7] checked whether numerical programs satisfy identity relations such as $P(x) = P(x_1) + P(x_2)$ for random values of x_1 and x_2. In the context of fault tolerance, the technique of data diversity [8] runs the program on re-expressed forms of the original input; e.g., $sin(x) = sin(a) \times sin(\pi/2 - b) + sin(\pi/2 - a) \times sin(b)$ where $a + b = x$. The concept of metamorphic testing, introduced by Chen [5] in 1998, generalises these ideas from identity relations to any type of relation, such as equalities, inequalities, periodicity properties, convergence constraints, subsumption relationships and many others. In general, a metamorphic relation for a function f is expressed as a relation among a series of function inputs $x_1, x_2, \ldots, x_n$ (with $n > 1$), and their corresponding output values $f(x_1), f(x_2), \ldots, f(x_n)$ [18]. For instance, for the sine example from the introduction the relation between x_1 and x_2 would be $\pi - x_1 = x_2$, and the relation between $f(x_1)$ and $f(x_2)$ would be equality, i.e.:

$$R = \{(x_1, x_2, \sin x_1, \sin x_2) \mid \pi - x_1 = x_2 \rightarrow \sin x_1 = \sin x_2\}$$

This resembles the traditional concept of program invariants, which are properties (for example expressed as assert statements) that hold at certain points in programs [19]. However, the key difference is that an invariant has to hold for every possible program execution, whereas a metamorphic relation is a relation between *different* executions. A relation between two executions implicitly defines how, given an existing source test case (x_1), one has to transform this into a follow-up test case (x_2), such that an abstract relation R (e.g., $\sin x_1 = \sin x_2$) can be checked on the inputs represented by x_1 and x_2, as well as the outputs produced by executing x_1 and x_2. The term metamorphic relation presumably refers to this "metamorphosis" of test inputs and outputs. If the relation R does not hold on a pair of source and follow–up test cases x_1 and x_2, then a fault has been detected. In this article, we use the term *metamorphic test case* to refer to a pair of a source test case and its follow–up test case.

The basic process for the application of metamorphic testing can be summarised as follows:

1) *Construction of metamorphic relations.* Identify necessary properties of the program under test and represent them as metamorphic relations among multiple test case inputs and their expected outputs, together with some method to generate a follow–up test case based on a source test case. Note that metamorphic relations may be associated with preconditions that restrict the source test cases to which they can be applied.
2) *Generation of source test cases.* Generate or select a set of source test cases for the program under test using any traditional testing technique (e.g., random testing).
3) *Execution of metamorphic test cases.* Use the metamorphic relations to generate follow–up test cases, execute source and follow–up test cases, and check the relations. If the outputs of a source test case and its follow–up test case violate the metamorphic relation, the metamorphic test case is said to have failed, indicating that the program under test contains a bug.

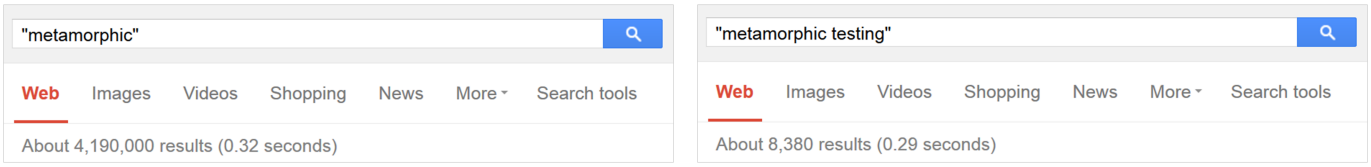
As an illustrative example, consider a program that computes the shortest path between a source vertex s and destination vertex d in a graph G, $SP(G, s, d)$. A metamorphic relation of the program is that if the source and destination vertices are swapped, the length of the shortest path should be equal: $|SP(G, s, d)| = |SP(G, d, s)|$. Suppose that a source test case (G, a, b) is selected according to some testing method (e.g., randomly). Based on the metamorphic relation, we can now easily generate a new follow–up test case by swapping the source and destination vertices (G, b, a). After executing the program with both test cases, their outputs can be checked against the relation to confirm whether it is satisfied or not, i.e., whether the outputs are equal. If the metamorphic relation is violated, it can be concluded that the metamorphic test has failed and the program is faulty.

As a further example, consider testing an online search engine such as Google or Yahoo [20]. Let $Count(q)$ be the number of results returned for a search query q. Intuitively, the number of returned results for q should be greater or equal than that obtained when refining the search with another keyword k. This can be expressed as the following metamorphic relation: $Count(q) \geq Count(q + k)$, where $+$ denotes the concatenation of two keywords. Fig. 1 illustrates the application of this metamorphic relation on Google. Consider a source test case consisting in a search for the keyword "metamorphic", resulting in "*About*" 4.2M results. Suppose that a follow–up test case is constructed by searching for the keywords "metamorphic testing": This leads to 8,380 results which is less than the result for "metamorphic", and thus satisfies the relation. If more results were found, then that would violate the metamorphic relation, revealing a bug in the system.

If source test cases are generated automatically, then metamorphic testing enables full test automation, i.e., input generation and output checking. In the sine example presented in Section 1, for instance, metamorphic testing could be used together with random testing to automatically generate random source test cases (x) and their respective follow–up test cases ($\pi - x$), until a pair is found that violates the metamorphic relation, or a maximum time out is reached. Similarly, in the search engine example, metamorphic testing could also be used together with a random word generator to automatically construct source test cases (e.g., "algorithm") and their respective follow–up test cases (e.g., "algorithm colour") until a pair that reveals a bug is found, if any such pairs exists.

3 REVIEW METHOD

To perform a literature review on metamorphic testing we followed a systematic and structured method inspired by the guidelines of Kitchenham [21] and Webster et al. [22]. A similar approach was followed by some of the authors in the context of software product lines [23]. To report the results, we also took inspiration from recent surveys on related topics such as the oracle problem [3], search–based testing [24], automated test case generation [2] and mutation



(a) Source test case. (b) Follow-up test case.

Figure 1. Metamorphic test on the Google search engine checking the relation $Count(q) \geq Count(q + k)$.

analysis [25]. Below, we detail the main data regarding the review process and its results.

3.1 Research questions

The aim of this literature review is to answer the following research questions on metamorphic testing:

- **RQ1**: *What improvements to the technique have been made?*
- **RQ2**: *What are its known application domains?*
- **RQ3**: *How are experimental evaluations performed?*
- **RQ4**: *What are the future research challenges?*

We propose RQ1 to obtain an in–depth view on metamorphic testing outlining the state of the art in terms of the main advances in the application of the technique since its original introduction. RQ2 is proposed to give an insight into the scope of metamorphic testing and its applicability to different domains including its integration with other testing techniques. We also want to know how different approaches of performing metamorphic testing are evaluated including the subject programs used, types of detected faults, evaluation metrics, and empirical studies involving humans. Finally, based on the answer to the previous questions, we expect to identify unresolved problems and research opportunities in response to RQ4.

3.2 Inclusion and exclusion criteria

We scrutinised the existing literature, looking for papers addressing any topic related to metamorphic testing, including methods, tools or guidelines for the application of the technique, applications to specific testing problems, empirical evaluations, and surveys. Articles of the same authors but with very similar content were intentionally classified and evaluated as separate contributions for a more rigorous analysis. Later, in the presentation of results, we grouped those articles with no major differences. We excluded PhD theses as well as those papers not related to the computer sciences field, not written in English, or not accessible on the Web.

3.3 Source material and search strategy

The search for relevant papers was carried out in the online repositories of the main technical publishers, including ACM, Elsevier, IEEE, Springer and Wiley. We collected computer science papers published between January 1st 1998 (when Chen's report was published) and November 30th 2015 which have either "metamorphic test", "metamorphic testing", "metamorphic relation" or "metamorphic relations" in their title, abstract or keywords. After a quick review of the results, we noticed that some articles on metamorphic testing with many citations were not among the candidate papers, including the technical report of Chen et al. [5] where the technique was introduced. To include those papers, we performed the search in the Google Scholar database, and additionally selected all papers with 5 or more citations published outside our target publication sources[2]. These were merged with our previous results, resulting in a final set of 362 candidate papers.

Next, we examined the abstracts of the papers identified in the previous step and filtered them according to our inclusion and exclusion criteria, checking the content of the papers when unsure. This step was performed by two different authors who agreed on the results. The set of candidate papers was filtered to 115 publications within the scope of our survey. Then, we contacted the corresponding authors of the 115 selected papers and asked them to inform us about any missing papers within the scope of our search. Based on the feedback received, we included 3 new papers meeting our search criteria, except for the inclusion of the search terms in their title, abstract or keywords. As a result, the search was finally narrowed to **118 publications** that were in the scope of this survey. These papers are referred to as the *primary studies* [21]. Table 1 presents the number of primary studies retrieved from each source.

It is possible that our search has failed to find all papers since we focused on a subset of reputed publishers. However, we remain confident that the overall trends we report are accurate and provide a fair picture of the state of the art on metamorphic testing.

3.4 Data collection

All 118 primary studies were carefully analysed to answer our research questions. For each study, we extracted the following information: full reference, brief summary, type of contribution (e.g., case study), application domains, integration with other testing techniques, number of metamorphic relations proposed, evaluation details, lessons learned and suggested challenges. To facilitate the process, we filled in a data extraction form for each primary study. All the forms are attached to this report in Appendix B.

Primary studies were read at least twice by two different authors to reduce misunderstandings or missing information. As a sanity check, we contacted the corresponding author of each primary study and sent them the technical report to confirm that the information collected from their papers was correct.

2. The search was performed on December 30th, 2015.

Table 1
Search engines used and number of results.

Search engine	Search queries	Results	Primary studies
ACM digital library	i) acmdlTitle:("metamorphic testing" "metamorphic test" "metamorphic relation" "metamorphic relations") 1998-2015, ii) recordAbstract:("metamorphic testing" "metamorphic test" "metamorphic relation" "metamorphic relations") 1998-2015, iii) keywords.author.keyword:("metamorphic testing" "metamorphic test" "metamorphic relation" "metamorphic relations") 1998-2015	12	12
Elsevier ScienceDirect	pub-date > 1997 and pub-date < 2015 and TITLE-ABSTR-KEY("metamorphic testing") or TITLE-ABSTR-KEY("metamorphic test") or TITLE-ABSTR-KEY("metamorphic relations") or TITLE-ABSTR-KEY("metamorphic relation")[All Sources(Computer Science)]	6	6
IEEEXplore digital library	((((((((((("Document Title":"metamorphic testing") OR "Abstract":"metamorphic testing") OR "Author Keywords":"metamorphic testing") OR "Document Title":"metamorphic test") OR "Abstract":"metamorphic test") OR "Author Keywords":"metamorphic test") OR "Document Title":"metamorphic relations") OR "Abstract":"metamorphic relations") OR "Author Keywords":"metamorphic relations") OR "Document Title":"metamorphic relation") OR "Abstract":"metamorphic relation" OR "Author Keywords":"metamorphic relation"))) and refined by Year: 1998-2015	72	65
Springer online library	"metamorphic testing" OR "metamorphic test" OR "metamorphic relation" OR "metamorphic relations"' within 1998 - 2014	87	13
Wiley InterScience	"metamorphic testing" in Article Titles OR "metamorphic testing" in Abstract OR "metamorphic testing" in Keywords OR "metamorphic test" in Article Titles OR "metamorphic test" in Abstract OR "metamorphic test" in Keywords OR "metamorphic relations" in Article Titles OR "metamorphic relations" in Abstract OR "metamorphic relations" in Keywords OR "metamorphic relation" in Article Titles OR "metamorphic relation" in Abstract OR "metamorphic relation" in Keywords NOT geology in All Fields NOT zoology in All Fields NOT ecology in All Fields between years 1998 and 2015	29	4
Google Scholar (+5 citations)	All of the words: "software", Any of the words: "metamorphic testing" "metamorphic test" "metamorphic relations" "metamorphic relation", None of the words: "zoology" "geology" "ecology", "anywhere in the article" Date filter: 1998-2015 (Citations filtered using the Publish or Perish program [26])	156	15

3.5 Summary of results

The following sections summarise the primary studies in terms of publication trends, authors, venues, and research topics on metamorphic testing.

3.5.1 Publication trends

Fig. 2a illustrates the number of publications on metamorphic testing published between January 1st 1998 and November 30th 2015. The graph shows a constant flow of papers on the topic since 2001, in particular from 2010 onwards. The cumulative number of publications is illustrated in Fig. 2b. We found a close fit to a quadratic function with a high determination coefficient ($R^2 = 0.998$), indicating a strong polynomial growth, a sign of continued health and interest in the subject. If the trend continues, there will be more than 160 metamorphic testing papers by 2018, two decades after the introduction of the technique.

3.5.2 Researchers and organisations

We identified 183 distinct co-authors from 74 different organisations in the 118 primary studies under review. Table 2 presents the top authors on metamorphic testing and their most recent affiliation. Unsurprisingly, Prof. T. Y. Chen, with 44 papers, is the most prolific author on the topic.

3.5.3 Geographical distribution of publications

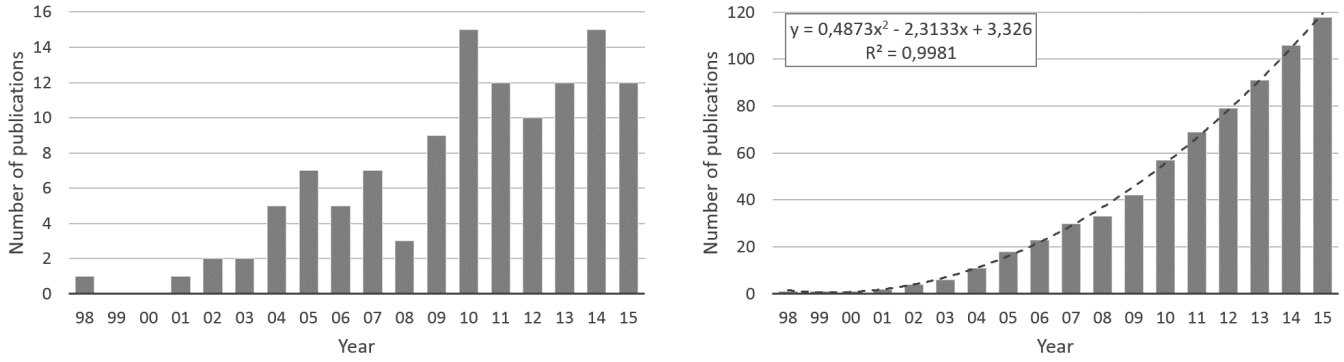
We related the geographical origin of each primary study to the affiliation country of its first co–author. Interestingly, we found that all 118 primary studies originated from only 11 different countries with Australia and China ahead, as presented in Table 3. By continents, 37% of the papers originated from Asia, 30% from Oceania, 19% from Europe and 14% from America. This suggests that the metamorphic testing community is formed by a modest number of countries but fairly distributed around the world.

3.5.4 Publication venues

The 118 primary studies under review were published in 72 distinct venues. This means that the metamorphic testing literature is very dispersed, probably due to its applicability to multiple testing domains. Regarding the type of venue, most papers were presented at conferences and symposia (58%), followed by journals (23%), workshops (16%) and technical reports (3%). Table 4 lists the venues where at least three metamorphic testing papers have been presented.

3.5.5 Types of contributions and research topics

Fig. 3a classifies the primary studies according to the type of contribution. We found that half of the papers present case studies (51%), followed by new techniques and methodologies (31%), and assessments and empirical studies (9%). We also found a miscellany of papers (7%) including related



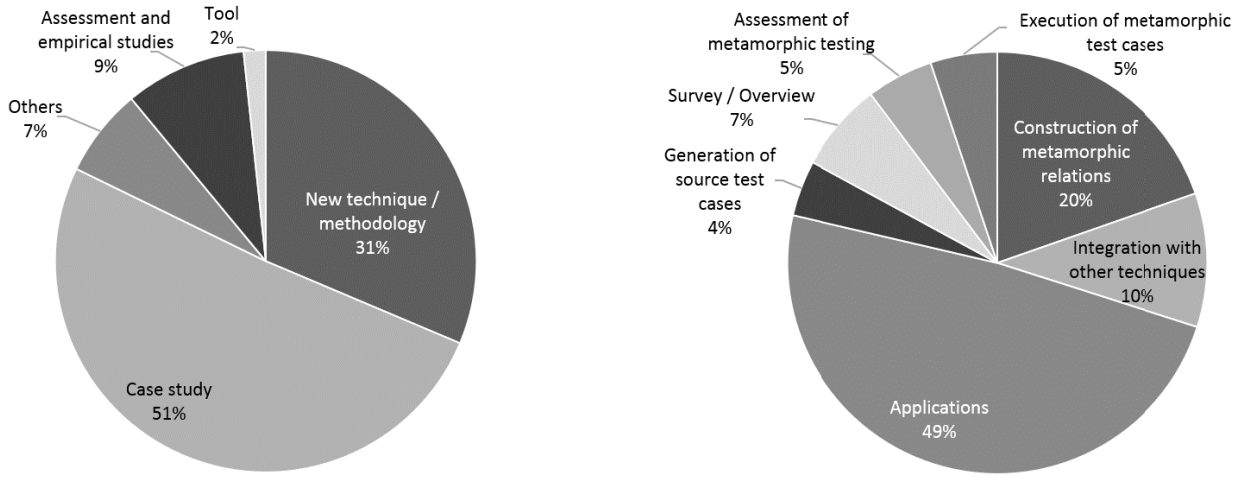
(a) Number of publications per year.

(b) Cumulative number of publications per year.

Figure 2. Metamorphic testing papers published between January 1st 1998 and November 30th 2015.

Table 2
Top 10 co-authors on metamorphic testing

Author	Institution	Country	Papers
T. Y. Chen	Swinburne University of Technology	Australia	44
T. H. Tse	The University of Hong Kong	Hong Kong	20
F.-C. Kuo	Swinburne University of Technology	Australia	17
Z. Q. Zhou	University of Wollongong	Australia	14
W. K. Chan	City University of Hong Kong	Hong Kong	11
H. Liu	RMIT University	Australia	9
C. Murphy	Columbia University	United States	8
G. Kaiser	Columbia University	United States	7
X. Xie	Swinburne University of Technology	Australia	7
B. Xu	Nanjing University	China	7



(a) Type of contribution.

(b) Research topic.

Figure 3. Classification of primary studies by publication type and research topic.

surveys, tutorial synopsis, and guidelines. Only two of the papers (2%) presented a tool as their main contribution.

A similar classification based on the main research topic is presented in Fig. 3b. Interestingly, we found that 49% of the papers report applications of metamorphic testing to different problem domains. The rest of papers address the construction of metamorphic relations (20%), integration with other testing techniques (10%), assessment of metamorphic testing (5%), execution of metamorphic test cases (5%) and generation of source test cases (4%). Finally, a few papers (7%) present brief overviews on the technique, its applications and research directions.

4 State of the Art in Metamorphic Testing

In this section, we address RQ1 by summarising the main contributions to metamorphic testing in the literature. First, we review the papers studying the properties of effective metamorphic relations. Then, approaches are classified according to the step they contribute to in the metamorphic

Table 3
Geographical distribution of publications

Country	Papers
Australia	36
China	25
United States	16
Hong Kong	12
Germany	8
Spain	7
India	5
United Kingdom	3
Switzerland	3
Malaysia	2
France	1

Table 4
Top venues on metamorphic testing.

Venue	Papers
Int Conference on Quality Software	9
Int Computer Software & Applications Conference	8
Int Workshop on Automation of Software Test	4
Int Conference on Software Engineering	4
IEEE Transactions on Software Engineering	4
Software Testing, Verification and Reliability	4
Int Conf on Software Testing, Verification and Validation	3
Information and Software Technology	3

testing process presented in Section 2, namely, construction of metamorphic relations, generation of source test cases, and execution of metamorphic test cases.

4.1 Properties of good metamorphic relations

The effectiveness of metamorphic testing is highly dependent on the specific metamorphic relations that are used, and designing effective metamorphic relations is thus a critical step when applying metamorphic testing. For most problems, a variety of metamorphic relations with different fault–detection capability can be identified [9], [16], [18], [27], [28], [29], [30], [31], [32], [33], [34], [35]. Therefore, it is advisable to use a variety of diverse metamorphic relations to effectively test a given program. Several authors even suggest using as many metamorphic relations as possible during testing [28], [29], [36], [37]. However, because defining metamorphic relations can be difficult, it is important to know how to select the most effective ones. In this section, we review papers studying the properties that make metamorphic relations *good* at detecting faults.

Defining good metamorphic relations requires knowledge of the problem domain. Chen et al. [27] compared the effectiveness of metamorphic relations solely based on the theoretical knowledge of the problem (black–box) versus those derived from the program structure (white–box) using two case studies. They concluded that theoretical knowledge of the problem domain is not adequate for distinguishing good metamorphic relations. Instead, good metamorphic relations should be preferably selected with regard to the algorithm under test following a white–box approach. However, this was later disputed by Mayer and Guderlei [38], who studied six subject programs for matrix determinant computation with seeded faults. They concluded that metamorphic relations in the form of equalities or linear equations[3] as well as those close to the implementation strategy have limited effectiveness. Conversely, they reported that good metamorphic relations are usually strongly inspired by the semantics of the program under test. Other studies have also emphasised the knowledge of the problem domain as a requirement for the application of metamorphic testing [30], [39], [40].

Metamorphic relations should make execution of the follow–up test case as different as possible from the source test case. Chen et al. [27] reported that good metamorphic relations are those that can make the execution of the source–test case as different as possible to its follow–up test case. They defined the "difference among executions" as any aspects of program runs (e.g., paths traversed). This observation has been confirmed by several later studies [9], [41], [42], [43], [44], [45]. In particular, Asrafi et al. [46] hypothesised that the higher the combined code coverage of the source and follow-up test cases, the more different are the executions, and the more effective is the metamorphic relation. Their study on two subject programs showed a strong correlation between coverage and fault–detection effectiveness in one of the two. In a similar study, Cao et al. [47] assessed the relation between fault–detection effectiveness of metamorphic relations and test case dissimilarity. An extensive experiment with 83 faulty programs and 7 distance metrics between the execution profiles of source and follow–up test cases revealed a strong and statistically significant correlation between the fault–detection capability of metamorphic relations and the distance among test cases, in particular when using branch coverage Manhattan distance [48].

Metamorphic relations derived from specific parts of the system are more effective than those targeting the whole system. Several authors have explored the applicability of metamorphic testing for integration testing with some helpful conclusions for the construction of good metamorphic relations. Just and Schweiggert [49], [50] assessed the applicability of metamorphic testing for system and integration testing in the context of an image encoder. Among other results, they concluded that the metamorphic relations derived from the components of a system are usually better at detecting faults than those metamorphic relations derived from the whole system. This finding was later confirmed by Xie et al. [51], who reported that metamorphic relations targeting specific parts of the program under test are easier to construct, more constrained, and therefore more effective in detecting faults than metamorphic relations at the system level.

Metamorphic relations should be formally described. Chan et al. [52] formally described metamorphic relations and metamorphic testing for a precise definition of the technique. Their formalisation was reused by several authors [29], [36] and later revised by Chan and Tse [12]. Hui and Huang [53] pointed out that most metamorphic

3. The authors literally refer to "equations with linear combinations on each side (with at least two terms on one of the sides)"

relations in the literature are informally described using natural language, which makes them easily misunderstood, ambiguous and hard to reuse. The authors suggested that good metamorphic relations should be formally described and proposed a formal model for the rigorous description of metamorphic relations using predicate logic, inspired by the work of Chan et al. [52]. In particular, they proposed representing a metamorphic relation as a 3–tuple composed of i) relation between the inputs of source and follow–up test cases, ii) relation between the outputs of source and follow-up test cases, and iii) program function.

4.2 Construction of metamorphic relations

Constructing metamorphic relations is typically a manual task that demands thorough knowledge of the program under test. In this section, we review proposed alternative ways to create metamorphic relations, either by combining existing relations, or by generating them automatically.

Liu et al. [54] proposed a method named *Composition of Metamorphic Relations (CMR)* to construct new metamorphic relations by combining several existing relations. A similar idea had been superficially explored previously by Dong et al. [55]. The rationale behind this method is that the resulting relations should embed all properties of the original metamorphic relations, and thus they should provide similar effectiveness with a fewer number of metamorphic relations and test executions. Intuitively, Liu et al. defined two metamorphic relations as *"compositable"* if the follow–up test cases of one of the relations can always be used as source test case of the other. The composition is sensitive to the order of metamorphic relations and generalisable to any number of them. Determining whether two metamorphic relations are composable is a manual task. The results of a case study with a bioinformatics program processing an input matrix show that the composition of a set of metamorphic relations usually produces a composite relation with higher (or at least similar) fault–detection effectiveness than the original metamorphic relations, provided that all component relations have similar "tightness". The tightness of a relation determines how hard it is to satisfy it by mere chance — the tighter a relation is, the more difficult it is to satisfy it with some random outputs; e.g., $\sin(x) = \sin(\pi - x)$ is tighter than $\sin(x) \neq \sin(\pi - x/2)$. They also concluded that the CMR method delivers higher cost–effectiveness than classic metamorphic testing since it involves fewer test executions.

Kanewala and Bieman [56], [57] proposed a method that determines, given a predefined set of relations that they believe to hold for many numerical programs, which of these are exhibited by a given numerical program. Their method works by extracting a function's control flow graph and building a predictive model using machine learning techniques; i.e., it is a white-box method that requires static access to the source code. The approach was evaluated by constructing a prediction model using a code corpus of 48 mathematical functions with numerical inputs and outputs. The model was designed to predict three specific types of metamorphic relations: permutative, additive and inclusive [58]. In addition, they checked the fault–detection effectiveness of the predictive metamorphic relations using seeded faults. The results revealed that 66% of the faults (655 out of 988) were detected by the predicted metamorphic relations. In later work [59], the authors extended their method using graph kernels, which provide various ways of measuring similarity among graphs. The intuition behind their approach was that functions that have similar control flow and data dependency graphs may have similar metamorphic relations. Empirical results on the prediction of six different types of metamorphic relations on a corpus of 100 numerical programs revealed that graph kernels lead to higher prediction accuracy.

Zhang et al. [60] proposed a search–based approach for the inference of polynomial metamorphic relations. More specifically, the algorithm searches for metamorphic relations in the form of linear or quadratic equations (e.g., $cos(2x) = 2cos^2(x) - 1$). Relations are inferred by running the program under test repeatedly, searching for relations among the inputs and outputs. It is therefore a black–box approach which requires no access to the source code. Since running the program with all the possible input values is rarely possible, the relations identified are strictly referred to as *likely* metamorphic relations, until they are confirmed by a domain expert. Their work was evaluated inferring hundreds of likely metamorphic relations for 189 functions of 4 commercial and open source mathematical libraries. The results showed that the generated metamorphic relations are effective in detecting mutants. Notice that in contrast to the work of Kanewala and Bieman [56], [57], this approach does not predict whether the program exhibits a previously defined metamorphic relation, but rather infers the metamorphic relation from scratch.

Carzinaga et al. [61] proposed to generate oracles by exploiting the redundancy contained in programs. Given a source test case, they generate a test with the same code in which some operations are replaced with redundant ones. For instance, in the `AbstractMultimap<K,V>` class of the Google Guava library[4], the methods `put(k,v)` and `putAll(k,c)` are equivalent when `c` is a collection containing a single element `v`. If the outputs of both test cases are not equal, the code must contain a bug. The author presented an implementation of their approach using aspects. The identification of redundant methods is a manual task. Although the core of their contribution was not related to metamorphic testing, their approach can be considered a specific application of the technique. In a related article, Goffi et al. [62], [63] presented a search–based algorithm for the automated synthesis of *likely–equivalent* method sequences in object–oriented programs. The authors suggest that such likely–equivalent sequences could be used as metamorphic relations during testing. The approach was evaluated using 47 methods of 7 classes taken from the Stack Java Standard Library and the Graphstream library. The algorithm automatically synthesised 87% (123 out of 141) of the equivalent method sequences manually identified.

Su et al. [64] presented an approach named KABU for the dynamic inference of likely metamorphic relations inspired by previous work on the inference of program invariants [19]. The inference process is constrained by searching for a set of predefined metamorphic relations [58]. A Java tool implementing the approach was presented and eval-

4. https://github.com/google/guava

uated on the inference of likely metamorphic relations in two sample programs. As a result, KABU found more likely metamorphic relations than a group of 23 students trained in the task. Authors also proposed a method, *Metamorphic Differential Testing (MDT)*, built upon KABU, to compare the metamorphic relations between different versions of the same program reporting the differences as potential bugs. Experimental results on different versions of two classification algorithms showed that MDT successfully detected the changes reported in the logs of the Weka library.

Chen et al. [65] presented a specification–based methodology and associated tool called METRIC for the identification of metamorphic relations based the category–choice framework [66]. In this framework, the program specification is used to partition the input domain in terms of categories, choices and complete test frames. Roughly speaking, a complete test frame is an abstract test case defining possible combinations of inputs, e.g., {*type of vehicle, weekday, parking hours*}. Given a set of complete test frames, METRIC guides testers on the identification of metamorphic relations and related source and follow-up test cases. The results of an empirical study with 19 participants suggest that METRIC is effective and efficient at identifying metamorphic relations.

4.3 Generation of source test cases

As mentioned in Section 6.2, most contributions on metamorphic testing use either random test data or existing test suites for the creation of source test cases. In this section, we review the papers proposing alternative methods for the generation of source test cases.

Gotlieb and Botella [67] presented a framework named *Automated Metamorphic Testing (AMT)* to automatically generate test data for metamorphic relations. Given the source code of a program written in a subset of C and a metamorphic relation, AMT tries to find test cases that violate the relation. The underlying method is based on the translation of the code into an equivalent constraint logic program over finite domains. The solving process is executed until a solution is found or a timeout is reached. The supported types of metamorphic relations are limited to numeric expressions over integers. The framework was evaluated using three laboratory programs with seeded faults.

Chen et al. [28] compared the effectiveness of "special values" and random testing as source test cases for metamorphic testing. Special values are test inputs for which the expected output is well known (e.g., $sin(\pi/2) = 1$). Since test cases with special values must be manually constructed we consider them as manual testing. The authors found that manual and metamorphic testing are complementary techniques, but they also note that random testing has the advantage of being able to provide much larger test data sets. In a closely related study, Wu et al. [68] concluded that random source test cases result in more effective metamorphic test cases than those derived from manual test cases (special values). Segura et al. [69] compared the effectiveness of random testing and a manually designed test suite as the source test cases for metamorphic testing, and their results also showed that random source test cases are more effective at detecting faults than manually designed source test cases in all the subject programs. Even though this suggests that random testing is more effective, there are also indications that *combining* random testing with manual tests may be even better: Chen et al [28] concluded that random testing is an efficient mechanism to augment the number of source test cases; Segura et al. [69] observed that combining manual tests with random tests leads to faster fault detection compared to using random tests only.

Batra and Sengupta [41] presented a genetic algorithm for the selection of source test cases maximising the paths traversed in the program under test. The goal is to generate a small but highly effective set of source test cases. Their algorithm was evaluated by generating source test cases for several metamorphic relations in a small C program, which determines the type of a triangle, where 4 mutants were generated and killed. In related work, Chen et al. [42] addressed the same problem from a black–box perspective. They proposed partitioning the input domain of the program under test into equivalence classes, in which the program is expected to process the inputs in a similar way. Then, they proposed an algorithm to select test cases that cover those equivalence classes. Evaluation on the triangle program suggests that their algorithm can generate a small set of test cases with high detection rate.

Dong and Zhang [44] presented a method for the construction of metamorphic relations and their corresponding source test cases using symbolic execution. The method first analyses the source code of the program to determine the symbolic inputs that cause the execution of each path. Then, the symbolic inputs are manually inspected and used to guide the construction of metamorphic relations that can exercise all the paths of the program. Finally, source test cases are generated by replacing the symbolic inputs by real values. As in previous work, the approach was evaluated using a small C program with seeded faults.

4.4 Execution of metamorphic test cases

The execution of a metamorphic test case is typically performed in two steps. First, a follow–up test case is generated by applying a transformation to the inputs of a source test case. Second, source and follow–up test cases are executed, checking whether their outputs violate the metamorphic relation. In this section, we present those articles that either propose a different approach for the execution of metamorphic test cases, or to automate part of the process.

Several papers have contributed to the execution and assessment of metamorphic test cases. Wu [70] presented a method named *Iterative Metamorphic Testing (IMT)* to systematically exploit more information from metamorphic tests, by applying metamorphic relations iteratively. In IMT, a sequence of metamorphic relations are applied in a chain style, by reusing the follow–up test case of each metamorphic relation as the source test case of the next metamorphic relation. A case study was presented with a program for sparse matrix multiplication and more than 1300 mutants. The results revealed that IMT detects more faults than classic metamorphic testing and special value testing. Dong et al. [71] presented an algorithm integrating IMT and program path analysis. The algorithm runs metamorphic tests iteratively until a certain path coverage

criterion is satisfied. Segura et al. [69], [72], [73] presented a metamorphic testing approach for the detection of faults in variability analysis tools. Their method is based on the iterative application of a small set of metamorphic relations. Each relation relates two input variability models and their corresponding set of configurations, (i.e., output). In practice, the process can generate an unlimited number of random test cases of any size. In certain domains, it was necessary to apply the metamorphic relations in a certain order. Their approach was proven effective in detecting 19 real bugs in 7 different tools.

Guderlei and Mayer [74] proposed *Statistical Metamorphic Testing (SMT)* for the application of metamorphic testing to non–deterministic programs. SMT does not consider a single execution, but is based on studying the statistical properties of multiple invocations to the program under test. The method works by generating two or more sequences of outputs by executing source and follow–up test cases. Then, the sequences of outputs are compared according to their statistical properties using statistical hypothesis tests. The applicability of the approach was illustrated with a single metamorphic relation on a subject program with seeded faults. In later work, Murphy et al. [75] successfully applied SMT to the detection of faults in a health care simulation program with non–deterministic time events.

Murphy et al. [76], [77] presented an extension of the Java Modelling Language (JML) [78] for the specification and runtime checking of metamorphic relations. Their approach extends the JML syntax to enable the specification of metamorphic properties, which are included in the Java source code as annotations. The extension was designed so it could express the typical metamorphic relations observed by the authors in the domain of machine learning [79]. Additionally, they presented a tool, named Corduroy, that pre–processes the specification of metamorphic relations and generates test code that can be executed using JML runtime assertion checking, ensuring that the relations hold during program execution. For the evaluation, they specified 25 metamorphic relations on several machine learning applications uncovering a few defects.

Murphy et al. [80] presented a framework named Amsterdam for the automated application of metamorphic testing. The tool takes as inputs the program under test and a set of metamorphic relations, defined in an XML file. Then, Amsterdam automatically runs the program, applies the metamorphic relations and checks the results. The authors argue that in certain cases slight variations in the outputs are not actually indicative of errors, e.g., floating point calculations. To address this issue, the authors propose the concept of *heuristic test oracles*, by defining a function that determines whether the outputs are "close enough" to be considered equals. An experimental evaluation with three machine learning applications was reported.

Ding et al. [43] proposed a method named *Self-Checked Metamorphic Testing (SCMT)* combining metamorphic testing and structural testing. SCMT checks the code coverage of source and follow-up test cases during test execution to evaluate the quality of metamorphic relations. It is assumed that the higher the coverage, the more effective the metamorphic relation. The test coverage data obtained may be used to refine test cases by creating, replacing or updating metamorphic relations and their test data. It is also suggested that unexpected coverage outcomes could help detect false–positive results, which they define as a metamorphic relation that holds despite the program being faulty. The approach was evaluated using a cellular image processing program with one seeded bug.

Zhu [81] presented JFuzz, a Java unit testing tool using metamorphic testing. In JFuzz, tests are specified in three parts, namely i) source test case inputs (x), ii) possible transformations on the test inputs ($y = \pi - x$), and iii) metamorphic relations implemented as code assertions ($\sin(x) = \sin(\pi - x)$). Once these elements are defined, the tool automatically generates follow-up test cases by applying the transformations to the source test inputs, it executes source and follow-up test cases, and checks whether the metamorphic relations are violated.

5 The application of metamorphic testing

In this section, we answer RQ2 by investigating the scope of metamorphic testing and its applications. In particular, we review applications of metamorphic testing to specific problem domains, and summarise approaches that use metamorphic testing to enhance other testing techniques.

5.1 Application domains

In this section, we review those papers where the main contribution is a case study on the application of metamorphic testing to specific testing problems (58 out of 118). Fig. 4 classifies these papers according to their application domain. In total, we identified more than 12 different application areas. The most popular domains are web services and applications (16%) followed by computer graphics (12%), simulation and modelling (12%) and embedded systems (10%). We also found a variety of applications to other fields (21%) such as financial software, optimisation programs or encryption programs. Each of the other domains is explored in no more than four papers, to date. Interestingly, we found that only 4% of the papers reported results in numerical programs, even though this seems to be the dominant domain used to illustrate metamorphic testing in the literature.

Fig. 5 shows the domains where metamorphic testing applications have been reported in chronological order. Domains marked with (T) were only explored theoretically. As illustrated, the first application of metamorphic testing was reported in the domain of numerical programs back in 2002. While in the subsequent years the potential applications of metamorphic testing were mainly explored at a theoretical level, there are applications in multiple domains from 2007 onwards. The rest of this section introduces the papers reporting results in each application domain.

5.1.1 Web services and applications

Chan et al. [82], [83] presented a metamorphic testing methodology for Service–Oriented Applications (SOA). Their method relies on the use of so-called *metamorphic services* to encapsulate the services under test, execute source and follow–up test cases and check their results. Similarly, Sun et al. [34], [84] proposed to manually derive metamorphic relations from the WSDL description of web services. Their

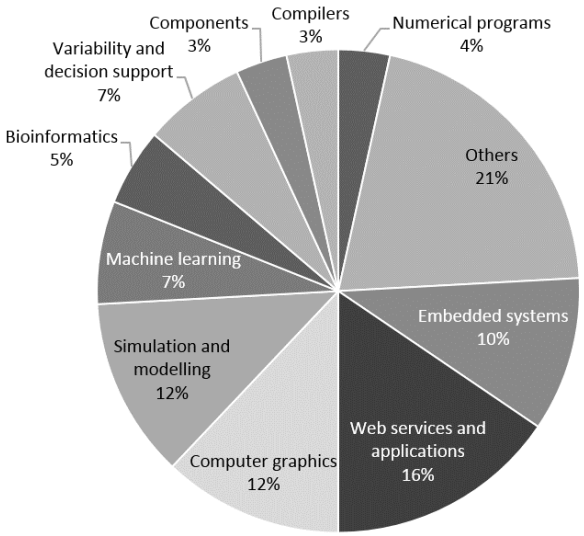


Figure 4. Metamorphic testing application domains

technique automatically generates random source test cases from the WSDL specification and applies the metamorphic relations. They presented a tool to partially automate the process, and evaluated it with three subject web services and mutation analysis. In a related project, Castro–Cabrera and Medina–Bulo [85], [86] presented a metamorphic testing–based approach for web service compositions using the Web Service Business Process Execution Language (WS–BPEL) [87]. To this end, they proposed to analyse the XML description of the service composition to select adequate metamorphic relations. Test cases were defined in terms of the inputs and outputs of the participant services.

In a related set of papers, Zhou et al. [20], [88] used metamorphic testing for the detection of inconsistencies in online web search applications. Several metamorphic relations were proposed and used in a number of experiments with the web search engines Google, Yahoo! and Live Search. Their results showed that metamorphic testing effectively detected inconsistencies in the searches in terms of both returned content and ranking quality. In later work [89], the authors performed an extensive empirical study on the web search engines Google, Bing, Chinese Bing and Baidu. As a novel contribution, metamorphic relations were defined from the user perspective, representing the properties that a user expects from a "good" search engine, regardless of how the engine is designed. In practice, as previously noticed by Xie et al. [31], this means that metamorphic relations are not only suitable to detect faults in the software under test (verification) but also to check whether the program behaves as the user expects (validation). The authors also proposed using metamorphic testing to assess quality related properties such as reliability, usability or performance. Experimental results revealed a number of failures in the search engines under test.

5.1.2 Computer graphics

Mayer and Guderlei [90], [91] compared several random image generation techniques for testing image processing programs. The study was performed on the implementation of several image operators as the Euclidean distance transform. Several metamorphic relations were used for the generation of follow–up test cases and the assessment of test results. Chan et al. [92], [93] presented a testing approach for mesh simplification programs using pattern classification and metamorphic testing. Metamorphic relations were used to detect test cases erroneously labelled as passed by a trained pattern classifier. Just and Schweiggert [94] used mutation analysis to evaluate the effectiveness of test data generation techniques and metamorphic relations for a jpeg2000 image encoder. Kuo et al. [33] presented a metamorphic testing approach for programs dealing with the surface visibility problem. A real bug was revealed in a binary space partitioning tree program. Finally, Jameel et al. [95] presented a case study on the application of metamorphic testing to detect faults in morphological image operations such as dilation and erosion. Eight metamorphic relations were reported and assessed on the detection of seeded faults in a binary image dilation program.

5.1.3 Embedded systems

Tse et al. [96] proposed the application of metamorphic testing to context–sensitive middleware–based software programs. Context–based applications adapt their behaviour according to the information from its environment referred to as *context*. The process of updating the context information typically relies on a *middleware*. Intuitively, their approach generates different context situations and checks whether the outcomes of the programs under test satisfy certain relations. This work was extended to deal with changes in the context during test execution [52], [97]. Chan et al. [98] applied metamorphic testing to wireless sensor networks. As a novel contribution, they proposed to check not only the functional output of source and follow–up test cases but also the energy consumed during the execution, thus targeting both functional and non–functional bugs. Kuo et al. [99] reported a case study on the use of metamorphic testing for the detection of faults in a wireless metering system. A metamorphic relation was identified and used to test the meter reading function of a commercial device from the electric industry in which two real defects were uncovered. Finally, Jiang et al. [100] presented several metamorphic relations for the detection of faults in Central Processing Unit (CPU) scheduling algorithms. Two real bugs were detected in one of the simulators under test.

5.1.4 Simulation and modelling

Sim et al. [101] presented an application of metamorphic testing for casting simulation, exploiting the properties of the Medial Axis geometry function. Several metamorphic relations were introduced but no empirical results were presented. Chen et al. [102] proposed the application of metamorphic testing to check the conformance between network protocols and network simulators. A case study was presented testing the OMNeT++ simulator [103] for conformance with the ad–hoc on–demand distance vector protocol. In a related project, Chen et al. [37] proposed using metamorphic testing for the detection of faults in open queuing network modelling, a technique for planning the capacity of computer and communication systems. Ding et al. [104] presented a case study on the detection of faults

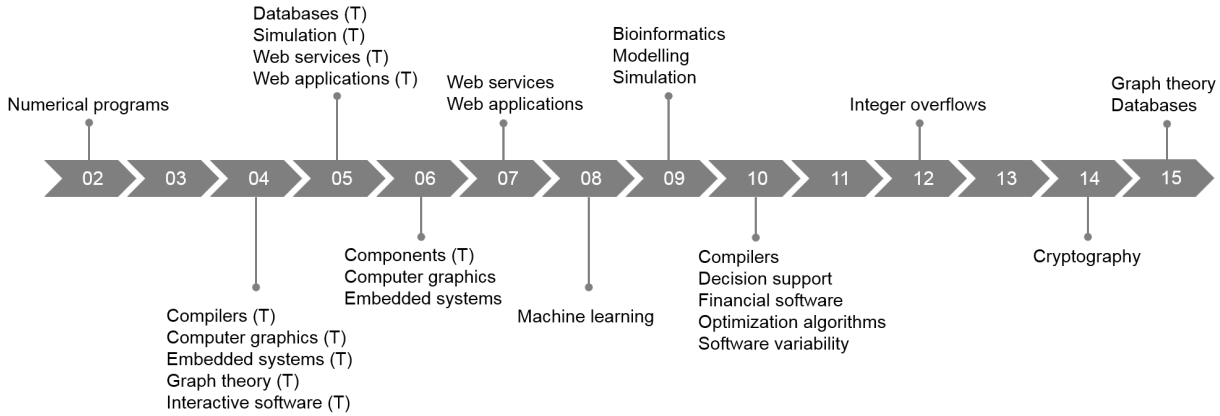


Figure 5. Timeline of metamorphic testing applications. Domains marked with (T) were only explored theoretically.

in a Monte Carlo modelling program for the simulation of photon propagation. Based on their previous work [43], the authors used code coverage criteria to guide the selection of effective metamorphic relations and the creation of test cases. Murphy et al. [75] proposed using metamorphic relations to systematically test health care simulation programs, and presented a case study with two real–world simulators and mutation testing. More recently, Núñez and Hierons [105] proposed using metamorphic relations to detect unexpected behaviour when simulating cloud provisioning and usage. A case study using two cloud models on the iCanCloud simulator [106] was reported. Cañizares et al. [107] presented some preliminary ideas on the use of simulation and metamorphic testing for the detection of bugs related to energy consumption in distributed systems as cloud environments.

5.1.5 Machine learning

Murphy et al. [58] identified six metamorphic relations that they believe exist in most machine learning applications, namely: additive, multiplicative, permutative, invertive, inclusive, and exclusive relations. The effectiveness of the relations was assessed on three specific machine learning tools in which some real bugs were detected. In a related project, Xie et al. [31], [108] proposed using metamorphic testing for the detection of faults in supervised classifiers. It was argued that metamorphic relations may represent both necessary and expected properties of the algorithm under test. Violation of necessary properties are caused by faults in the algorithm and therefore are helpful for the purpose of verification. Violations of expected properties indicate divergences between what the algorithm does and what the user expects, and thus are helpful for the purpose of validation. Two specific algorithms were studied: K-Nearest neighbours and Naïve Bayes classifier. The results revealed that both algorithms violated some of the necessary properties identified as metamorphic relations indicating faults or unexpected behaviours. Also, some real faults were detected in the open–source machine learning tool Weka [109]. Finally, Jing et al. [110] presented a set of metamorphic relations for association rule algorithms and evaluated them using a contact–lenses data set and the Weka tool.

5.1.6 Variability and decision support

Segura et al. [69], [72] presented a test data generator for feature model analysis tools. Test cases are automatically generated from scratch using step–wise transformations that ensure that certain constraints (metamorphic relations) hold at each step. In later work [73], the authors generalised their approach to other variability domains, namely CUDF documents and Boolean formulas. An extensive evaluation of effectiveness showed, among other results, fully automatic detection of 19 real bugs in 7 tools. In a related domain[5], Kuo et al. [45] presented a metamorphic testing approach for the automated detection of faults in decision support systems. In particular, they focused on the so–called multi–criteria group decision making, in which decision problems are modelled as a three–dimensional matrix representing alternatives, criteria and experts. Several metamorphic relations were presented and used to test the research tool Decider [45], where a bug was uncovered.

5.1.7 Bioinformatics

Chen et al. [40] presented several metamorphic relations for the detection of faults in two open–source bioinformatics programs for gene regulatory networks simulations and short sequence mapping. Also, the authors discussed how metamorphic testing could be used to address the oracle problem in other bioinformatics domains such as phylogenetic, microarray analysis and biological database retrieval. Pullum and Ozmen [111] proposed using metamorphic testing for the detection of faults in predictive models for disease spread. A case study on the detection of faults in two disease–spread models of the 1918 Spanish flu was presented, revealing no bugs. In a related project, Ramanathan et al. [112] proposed using metamorphic testing, data visualisation, and model checking techniques to formally verify and validate compartmental epidemiological models.

5.1.8 Components

Beydeda [113] proposed a self–testing method for commercial off–the–shelf components using metamorphic testing. In this method, components are augmented with self–testing

5. Note that variability models can be used as decision models during software configuration.

functionality including test case generation, execution and evaluation. In practice, this method allows users of a component to test it even without access to its source code. Lu et al. [114] presented a metamorphic testing methodology for component–based software applications, both at the unit and integration level. The underlying idea is to run test cases against the interfaces of the components under test, using metamorphic relations to construct follow–up test cases and to check their results.

5.1.9 Numerical programs

Chen et al. [115] presented a case study on the application of metamorphic testing to programs implementing partial differential equations. The case study focused on a practical problem in thermodynamics, namely the distribution of temperatures in a square plate. They injected a seeded fault in the program under test and compared the effectiveness of "special" test cases and metamorphic testing in detecting the fault. Special test cases were unable to detect the fault, while metamorphic testing was effective at revealing it using a single metamorphic relation. Aruna and Prasad [116] presented several metamorphic relations for multiplication and division of multi–precision arithmetic software applications. The work was evaluated with four real–time mathematical projects and mutation analysis.

5.1.10 Compilers

Tao et al. [117] presented a so–called "equivalence preservation" metamorphic relation to test compilers. Given an input program, the relation is used to generate an equivalent variant of it, checking whether the behaviours of the resulting executables are the same for a random set of inputs. The authors proposed three different strategies for the generation of equivalent source programs, such as replacing an expression with an equivalent one (e.g., $e \times 2 \equiv e + e$). The evaluation of their approach revealed two real bugs in two C compilers. A closely related idea was presented by Le et al. [118]. Given a program and a set of input values, the authors proposed to create equivalent versions of the program by profiling its execution and pruning unexecuted code. Once a program and its equivalent variant are constructed, both are used as input of the compiler under test, checking for inconsistencies in their results. So far, this method has been used to detect 147 confirmed bugs in two open source C compilers, GCC and LLVM.

5.1.11 Other domains

Zhou et al. [39] presented several illustrative applications of metamorphic testing in the context of numerical programs, graph theory, computer graphics, compilers and interactive software. Chen et al. [119] claimed that metamorphic testing is both practical and effective for end–user programmers. To support their claim, the authors briefly suggested how metamorphic relations could be used to detect bugs in spreadsheet, database and web applications. Sim et al. [120] presented a metamorphic testing approach for financial software. Several metamorphic relations were integrated into the commercial tool MetaTrader [121] following a self–testing strategy. Source and follow-up test cases were derived from the real–time input price data received at different time periods. Metamorphic testing has also been applied to optimisation programs using both stochastic [122] and heuristic algorithms [32]. Yao et al. [123], [124], [125] presented preliminary results on the use of metamorphic testing to detect integer overflows. Batra and Singh [126] proposed using UML diagrams to guide the selection of metamorphic relations and presented a small case study using a banking application. Sun et al. [127] reported several metamorphic relations for encryption programs. Aruna and Prasad [128] presented a small case study on the application of metamorphic testing to two popular graph theory algorithms. Finally, Lindvall et al. [129] presented an experience report on the use of metamorphic testing to address acceptance testing of NASA's Data Access Toolkit (DAT). DAT is a huge database of telemetry data collected from different NASA missions, and an advance query interface to search and mine the available data. Due to the massive amount of data contained in the database, checking the correctness of the query results is challenging. To address this issue, metamorphic testing was used by formulating the same query in different equivalent ways, and asserting that the resulting datasets are the same. Several issues were detected with this approach.

5.2 Other testing applications

Besides direct application as a testing technique, metamorphic testing has been integrated into other testing techniques, in order to improve their applicability and effectiveness. In this section, we review these approaches.

Chen et al. [18], [130] proposed using metamorphic testing with fault–based testing. Fault–based testing uses symbolic evaluation [131], [132] and constraint solving [132] techniques to prove the absence of certain types of faults in the program under test. The authors used several numerical programs to illustrate how real and symbolic inputs can be used to discard certain types of faults even in the absence of an oracle. In a related project [30], [133], the authors presented a method called *semi–proving* integrating global symbolic execution and constraint solving for program proving, testing and debugging. Their method uses symbolic execution to prove whether the program satisfies certain metamorphic relations or identify the inputs that violate them. It also supports debugging by identifying violated constraint expressions that reveal failures.

Dong et al. [134] proposed improving the efficiency of Structural Evolutionary Testing (SET) using metamorphic relations. In SET, evolutionary algorithms are used to generate test data that satisfy a certain coverage criteria (e.g., condition coverage). This is often achieved by minimising the distance of the test input to execute the program conditions in the desired way. To improve the efficiency of the process, the authors proposed to use metamorphic relations during the search to consider both source and follow–up test cases as candidate solutions, accelerating the chances of reaching the coverage target. Their approach was evaluated with two numerical programs.

Xie et al. [135], [136] proposed the combination of metamorphic testing and Spectrum-Based Fault Localisation (SBFL) for debugging programs without an oracle. SBFL uses the results of test cases and the corresponding coverage information to estimate the risk of each program entity

(e.g., statements) of being faulty. Rather than a regular test oracle, the authors proposed to use the violation or non-violation information from metamorphic relations rather than the actual output of test cases. Among other results, their approach was used to uncover two real bugs in the Siemens Suite [137]. In a related project, Lei et al. [138] applied the same idea to address the oracle problem in a variant of SBFL named Backward-Slice Statistical Fault Localisation (BSSFL) [139]. Rao et al. [140] investigated the ratio between non-violated and violated metamorphic relations in SBFL. They concluded that the higher the ratio of non–violated metamorphic relations to violated metamorphic relations, the less effective the technique. Aruna et al. [141] proposed integrating metamorphic testing with the Ochiai algorithm [142] for fault localisation in dynamic web applications. Five metamorphic relations for a classification algorithm were presented as well as some experimental results.

Liu et al. [143] presented a theoretical description of a new method called *Metamorphic Fault Tolerance (MFT)*. In MFT, the trustworthiness of test inputs is determined in terms of the number of violated and non–violated metamorphic relations. The more relations are satisfied and the fewer relations are violated, the more trustworthy the input is. Also, if an output is judged as untrustworthy, the outputs provided by metamorphic relations can be used to provide a more accurate output.

Jin et al. [144] presented an approach called *Concolic Metamorphic Debugging*, which integrates concolic testing, metamorphic testing, and branch switching debugging, in order to localise potential bugs. Concolic testing is a technique that executes the program under test with both, symbolic and concrete inputs, and then uses symbolic path conditions to derive new test inputs for paths not yet explored. Based on a failure-inducing test input, the proposed method explores all possible program paths in depth-first-order, searching for the first one that passes the metamorphic relation. The final goal is to isolate a minimum amount of code to obtain a passing input, and use that isolation point to localise the fault. The approach, implemented in a tool called Comedy, was evaluated on 21 small programs with seeded faults. Comedy successfully generated debugging reports in 88% of the faulty programs and precisely located the fault in 36% of them.

6 Experimental evaluations

In this section, we address RQ3 by reviewing the experimental evaluations of the surveyed papers. In particular, we summarise their main characteristics in terms of subject programs, source test cases, types of faults, number of metamorphic relations and evaluation metrics. Additionally, we review the results of empirical studies involving humans.

6.1 Subject programs

As a part of the review process, we collected information about the subject programs used for the evaluation of metamorphic testing contributions. Appendix A shows the name, language, size, description and the references of the papers reporting results for each program. In the cases where the information was unavailable in the literature, it is indicated with "NR" (Not Reported). The table is ordered by the number of papers that use the subject programs. Thus, the programs at the top of the list are the most studied subject programs in the metamorphic testing literature. Overall, we identified 143 different subject programs. Most of them are written in Java (46.1%) and C/C++ (35.6%), with reported sizes ranging between 12 and 12,795 lines of code.

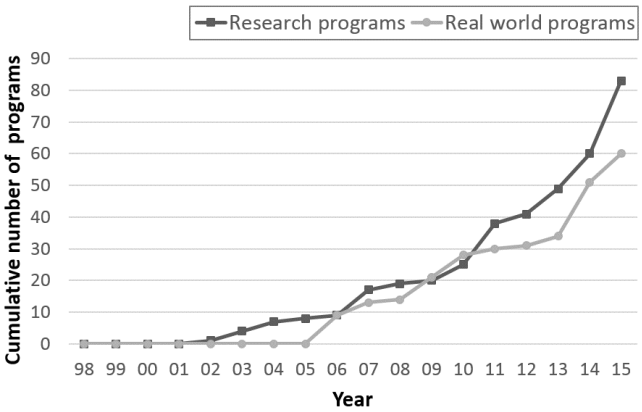


Figure 6. Research vs real world subject programs

In experimentation, the use of real world programs, rather than research programs, is commonly recognised as an indicator of the maturity of a discipline [25]. To assess this maturity, we studied the relationship between the use of research and real world programs in metamorphic testing experiments. Similarly to previous surveys [25], we consider a program to be a "real world" program if it is either a commercial or an open–source program, otherwise we consider it as a "research program". As an exception to this rule, we consider all open source projects that are designed as benchmarks rather than applications as research programs (e.g., the Siemens suite). Fig. 6 presents the cumulative view of the number of each type of program, research and real world, by year. As illustrated, research programs are used since 2002, while real world programs were not introduced in metamorphic testing experiments until 2006. Since then, the use of both types of programs has increased with similar trends. It is noteworthy that the number of real world programs in 2010 was higher than the number of research programs. The cumulative number in 2015 shows a significant advantage of research programs (83) over real world programs (60). The overall trend, however, suggests that metamorphic testing is maturing.

6.2 Source test cases

Metamorphic testing requires the use of source test cases that serve as seed for the generation of follow–up test cases. Source test cases can be generated using any traditional testing techniques. We studied the different techniques used in the literature and counted the number of papers using each of them; the results are presented in Fig. 7. As illustrated, a majority of studies used random testing for the generation of source test cases (58%), followed by those using an existing test suite (33%). Also, several papers (6%)

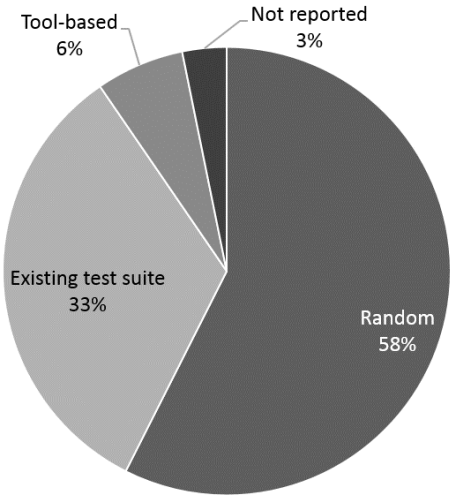


Figure 7. Source test case generation techniques

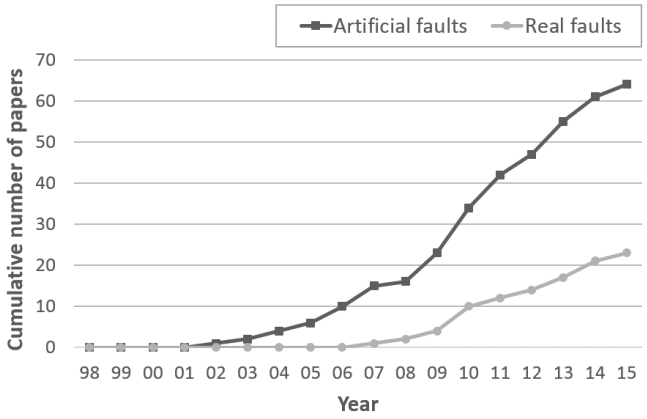


Figure 8. Artificial vs real faults

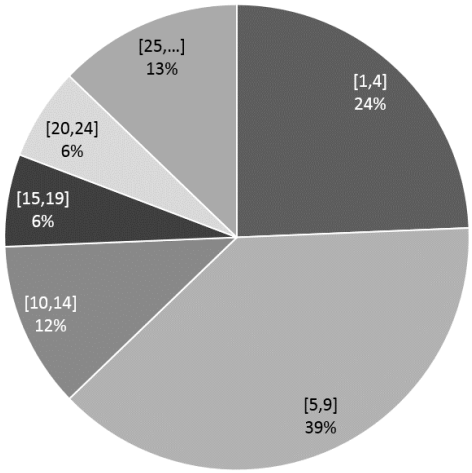


Figure 9. Number of metamorphic relations

use tool–based techniques such as constraint programming, search–based testing or symbolic execution. This diversity of usable sources supports the applicability of metamorphic testing. It also supports the use of random testing as a cost–effective and straightforward approach for the generation of the initial test suite (cf. Section 4.3).

6.3 Types of faults

The effectiveness of metamorphic testing approaches is assessed according to their ability to detect failures caused by faults in the programs under test. Uncovering real bugs is the primary goal, but they are not always available for evaluation. Thus, most authors introduce artificial faults (a.k.a. mutants) in the subject programs either manually or automatically, using mutation testing tools [25]. To study the relationship between real bugs and mutants in metamorphic testing evaluations, we calculated the cumulative number of papers reporting results with artificial and real bugs by year, depicted in Fig. 8. We consider a real bug to be a latent, initially unknown, fault in the subject program. As illustrated in Fig. 8, the first experimental results with mutants were presented back in 2002, while the first real bugs were reported in 2007. Since then, the number of papers reporting results with both types of faults has increased, although artificial faults show a steeper angle representing a stronger trend. Besides this, we also counted the number of faults used in each paper. To date, metamorphic testing has been used to detect about 295 distinct real faults in 36 different tools, 23 of which are real world programs, suggesting that metamorphic testing is effective at detecting real bugs.

6.4 Metamorphic relations

The number of metamorphic relations used in experimentation may be a good indicator of the effort required to apply metamorphic testing. As a part of the data collection process, we counted the number of metamorphic relations presented in each paper containing experimental results. Fig. 9 classifies the papers based on the number of metamorphic relations reported. As illustrated, the largest portion of studies report between 5 and 9 metamorphic relations (39%), followed by those presenting between 1 and 4 metamorphic relations (24%) and those reporting between 10 and 14 metamorphic relations (12%). Interestingly, only 9 studies (13%) presented more than 25 metamorphic relations. We took a closer look at those 9 papers and observed that all of them reported results for several subject programs. These findings suggest that a modest number of metamorphic relations (less than 10) is usually sufficient to apply metamorphic testing with positive results.

6.5 Evaluation metrics

Numerous metrics to evaluate the effectiveness of metamorphic testing approaches have been proposed. Among them, we identified two metrics intensively used in the surveyed papers, such that they could be considered as a de–facto standard in the metamorphic testing literature.

6.5.1 Mutation score

This metric is based on mutation analysis, where mutation operators are applied to systematically produce versions of the program under test containing artificial faults

("mutants") [25]. The mutation score is the ratio of detected ("killed") mutants to the total number of mutants. Mutants that do not change the program's semantics and thus cannot be detected are referred to as *equivalent* [25]. In theory, equivalent mutants should be excluded from the total number of mutants, but in practice this is not always possible since program equivalence is undecidable. Suppose a metamorphic test suite t composed of a set of metamorphic tests, i.e., pairs of source and follow–up test cases. The Mutation Score (MS) of t is calculated as follows:

$$MS(t) = \frac{M_k}{M_t - M_e} \quad (1)$$

where M_k is the number of killed mutants by the metamorphic tests in t, M_t is the total number of mutants and M_e is the number of equivalent mutants. A variant of this metric [71], [90], [120] is often used to calculate the ratio of mutants detected by a given metamorphic relation r as follows:

$$MS(t, r) = \frac{M_{kr}}{M_t - M_e} \quad (2)$$

where M_{kr} is the number of mutants killed by the metamorphic tests in t derived from r. This metric is also called mutation detection ratio [36].

6.5.2 Fault detection ratio

This metric calculates the ratio of test cases that detect a given fault [41], [55], [68], [70], [71], [100], [123], [125]. The Fault Detection Ratio (FDR) of a metamorphic test suite t and a fault f is calculated as follows:

$$FDR(t, f) = \frac{T_f}{T_t} \quad (3)$$

where T_f is the number of tests that detect f and T_t is the total number of tests in t. A variant of this metric [27], [32], [33], [37], [45], [54], [71] calculates the ratio of test cases that detect a fault f using a given metamorphic relation r as follows:

$$FDR(t, f, r) = \frac{T_{fr}}{T_r} \quad (4)$$

where M_{fr} is the number of tests in t derived from the relation r that detect the fault f, and T_r is the total number of metamorphic tests derived from r. This metric is also called fault discovery rate [34], [84], [127].

6.6 Empirical studies with humans

Hu et al. [29], [36] reported on a controlled experiment to investigate the cost–effectiveness of using metamorphic testing by 38 testers on three open–source programs. The experiment participants were either asked to write metamorphic relations, or tests with assertions to check whether the final or intermediate state of the program under test is correct. The experiment revealed a trade–off between both techniques, with metamorphic testing being less efficient but more effective at detecting faults than tests with assertions.

Liu et al. [145] reported on a 3–year experience in teaching metamorphic testing to various groups of students at Swinburne University of Technology (Australia). The authors explained the teaching approach followed and the lesson learned, concluding that metamorphic testing is a suitable technique for end–user testing. In a later paper, Liu et al. [4] presented an empirical study to investigate the effectiveness of metamorphic testing addressing the oracle problem compared with random testing. For the study, several groups of undergraduate and postgraduate students from two different universities were recruited to identify metamorphic relations in five subject programs of algorithmic type. Metamorphic testing was compared to random testing with and without oracle. Their experiment showed that metamorphic testing was able to find more faults than random testing with and without oracle in most subject programs. Furthermore, it was concluded that a small number of diverse metamorphic relations (between 3 and 6), even those identified in an ad–hoc manner, had a similar fault-detection capability to a test oracle, i.e., comparing the program output with the expected one.

7 Challenges

A number of open research challenges emerge from this literature review, based on problems repeatedly encountered throughout the reviewed papers, or gaps in the literature. These challenges answer RQ4.

Challenge 1: Guidelines for the construction of good metamorphic relations. For most problems, a variety of metamorphic relations with different fault–detection capability can be identified. It is therefore key to know the properties of effective metamorphic relations and to provide systematic methods for their construction. Although several authors have reported lessons learned on the properties of good metamorphic relations (cf. Section 4.1), these are often complementary or even contradictory (e.g., [27], [38]). Therefore, there is a lack of reliable guidelines for the construction of effective metamorphic relations. Such guidelines should provide a step–by–step process to guide testers, both experts and beginners, in the construction of good metamorphic relations.

Challenge 2: Prioritisation and minimisation of metamorphic relations. In certain cases using all the available metamorphic relations may be too expensive and a subset of them must be selected. It is therefore important to know how to prioritise the most effective metamorphic relations. To this end, several authors have proposed using code coverage [43], [46] or test case similarity [47] with promising results. However, the applicability of those approaches as domain–independent prioritisation criteria still needs to be explored. Furthermore, analogously to the concept of test suite minimisation, where redundant test cases are removed from a suite as it evolves [146], the use of minimisation techniques to remove redundant metamorphic relations is an open problem where research is needed. It is worth mentioning that test case minimisation is a NP–hard problem and therefore heuristic techniques should be explored.

Challenge 3: Generation of likely metamorphic relations. The generation of metamorphic relations is probably the

most challenging problem to be addressed. Although some promising results have been reported, those are mainly restricted to the scope of numerical programs. The generation of metamorphic relations in other domains as well as the use of different techniques for rule inference are topics where contributions are expected. We also foresee a fruitful line of research exploring the synergies between the problem of generating metamorphic relations and the detection of program invariants [64], [147].

Challenge 4: Combination of metamorphic relations. As presented in Section 4.2, several authors have explored the benefits of combining metamorphic relations following two different strategies, namely applying metamorphic relations in a chain style (IMT) and composing metamorphic relations to construct new relations (CMR). It remains an open problem, however, to compare both approaches and to provide heuristics to decide when to use one or the other. Also, these techniques raise new research problems such us determining whether a given set of metamorphic relations can be combined and in which order.

Challenge 5: Automated generation of source test cases. As described in Section 4.3, most papers use either randomly generated or existing test suites as source tests when applying metamorphic testing. However, there is evidence that the source test cases influence the effectiveness of metamorphic relations [28], [68], [69]. Promising initial results in generating source test cases specifically for given metamorphic relations have been achieved, but many open questions remain about what constitutes the best possible source test cases and how to generate them.

Challenge 6: Metamorphic testing tools. Only two out of all 118 presented a tool as main contribution [77], [81], and very few of the papers on metamorphic testing mentioned a tool implementing the presented techniques [64], [65], [67], [73], [80], [88], [117], [144]. Indeed, if practitioners want to apply metamorphic testing today, they would have to implement their own tool, as there are no publicly available and maintained tools. This is a significant obstacle for a wide-spread use of metamorphic testing in empirical research as well as in practice.

8 Conclusions

In this technical report, we presented a literature review on metamorphic testing covering 118 papers published between 1998 and 2015. We analysed ratios and trends indicating the main advances on the technique, its application domains and the characteristics of experimental evaluations. The results of the survey show that metamorphic testing is a thriving topic with an increasing trend of contributions on the subject. We also found evidence of the applicability of the technique to multiple domains far beyond numerical programs, as well as its integration with other testing techniques. Furthermore, we identified an increasing number of papers reporting the detection of faults in real world programs. All these findings suggest that metamorphic testing is gaining maturity as an effective testing technique, not only to alleviate the oracle problem, but also for the automated generation of test data. Finally, despite the advances on metamorphic testing, our survey points to areas where research is needed. We trust that this work may become a helpful reference for future development on metamorphic testing as well as to introduce newcomers in this promising testing technique.

Acknowledgements

We would like to thank T. Y. Chen, Robert M. Hierons, Phil McMinn, Amador Durán, Zhi Quan Zhou, Christian Murphy, Huai Liu, Xiaoyuan Xie, Alberto Goffi, Gagandeep, Carmen Castro-Cabrera, Yan Lei and Peng Wu for their helpful comments in an earlier version of this article. We are also grateful to the members of the SSE research group led by Mark Harman for the insightful and inspiring discussion during our visit at the University College London.

This work has been partially supported by the European Commission (FEDER) and Spanish Government under CICYT project TAPAS (TIN2012-32273) and the Andalusian Government projects THEOS (TIC-5906) and COPAS (P12-TIC-1867).

References

[1] E. J. Weyuker, "On testing non-testable programs," *The Computer Journal*, vol. 25, no. 4, pp. 465–470, 1982.

[2] S. Anand, E. K. Burke, T. Y. Chen, J. Clark, M. B. Cohen, W. Grieskamp, M. Harman, M. J. Harrold, and P. Mcminn, "An orchestrated survey of methodologies for automated software test case generation," *Journal of Systems and Software*, vol. 86, no. 8, pp. 1978–2001, Aug. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2013.02.061

[3] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey," *IEEE Transactions on Software Engineering*, vol. PP, no. 99, pp. 1–1, 2014.

[4] H. Liu, F.-C. Kuo, D. Towey, and T. Y. Chen, "How effectively does metamorphic testing alleviate the oracle problem?" *Software Engineering, IEEE Transactions on*, vol. 40, no. 1, pp. 4–22, Jan 2014.

[5] T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic testing: A new approach for generating next test cases," Technical Report HKUST-CS98-01, Department of Computer Science, The Hong Kong University of Science and Technology, Tech. Rep., 1998.

[6] W. J. Cody, *Software Manual for the Elementary Functions*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1980.

[7] M. Blum, M. Luby, and R. Rubinfeld, "Self-testing/correcting with applications to numerical problems," *Journal of Computer and System Sciences*, vol. 47, no. 3, pp. 549 – 595, 1993. [Online]. Available: http://www.sciencedirect.com/science/article/pii/002200009390044W

[8] P. E. Ammann and J. C. Knight, "Data diversity: An approach to software fault tolerance," *IEEE Transactions on Computers*, vol. 37, no. 4, pp. 418–425, Apr. 1988. [Online]. Available: http://dx.doi.org/10.1109/12.2185

[9] T. Y. Chen, F.-C. Kuo, T. H. Tse, and Z. Q. Zhou, "Metamorphic testing and beyond," in *Eleventh Annual International Workshop on Software Technology and Engineering Practice, 2003.*, Sept 2003, pp. 94–100.

[10] T. H. Tse, "Research directions on model-based metamorphic testing and verification," in *29th Annual International Computer Software and Applications Conference, 2005. COMPSAC 2005.*, vol. 1, July 2005, pp. 332 Vol. 2–.

[11] T. Y. Chen, "Metamorphic testing: A simple approach to alleviate the oracle problem," in *Fifth IEEE International Symposium on Service Oriented System Engineering (SOSE), 2010*, June 2010, pp. 1–2.

[12] W. K. Chan and T. H. Tse, "Oracles are hardly attain'd, and hardly understood: Confessions of software testing researchers," in *13th International Conference on Quality Software (QSIC), 2013*, July 2013, pp. 245–252.

[13] T. Y. Chen, "Metamorphic testing: A simple method for alleviating the test oracle problem," in *Proceedings of the 10th International Workshop on Automation of Software Test*, ser. AST '15. Piscataway, NJ, USA: IEEE Press, 2015, pp. 53–54. [Online]. Available: http://dl.acm.org/citation.cfm?id=2819261.2819278

[14] T. Y. Chen, F.-C. Kuo, D. Towey, and Z. Q. Zhou, "Metamorphic testing: Applications and integration with other methods: Tutorial synopsis," in *12th International Conference on Quality Software (QSIC), 2012*, Aug 2012, pp. 285–288.

[15] Z. Hui and S. Huang, "Achievements and challenges of metamorphic testing," in *ourth World Congress on Software Engineering (WCSE), 2013*, Dec 2013, pp. 73–77.

[16] U. Kanewala and J. M. Bieman, "Techniques for testing scientific programs without an oracle," in *5th International Workshop on Software Engineering for Computational Science and Engineering (SECSE), 2013*, May 2013, pp. 48–57.

[17] G. Dong, B. Xu, L. Chen, C. Nie, and L. Wang, "Survey of metamorphic testing," *Journal of Frontiers of Computer Science and Technology*, vol. 3, no. 2, pp. 130–143, 2009.

[18] T. Y. Chen, T. H. Tse, and Z. Q. Zhou, "Fault-based testing without the need of oracles," *Information & Software Technology*, vol. 45, no. 1, pp. 1–9, 2003. [Online]. Available: http://dx.doi.org/10.1016/S0950-5849(02)00129-5

[19] M. D. Ernst, J. H. Perkins, P. J. Guo, S. McCamant, C. Pacheco, M. S. Tschantz, and C. Xiao, "The daikon system for dynamic detection of likely invariants," *Sci. Comput. Program.*, vol. 69, no. 1-3, pp. 35–45, Dec. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.scico.2007.01.015

[20] Z. Q. Zhou, S. Zhang, M. Hagenbuchner, T. H. Tse, F.-C. Kuo, and T. Y. Chen, "Automated functional testing of online search services," *Software Testing, Verification and Reliability Journal*, vol. 22, no. 4, pp. 221–243, Jun. 2012. [Online]. Available: http://dx.doi.org/10.1002/stvr.437

[21] B. Kitchenham, "Procedures for performing systematic reviews," Keele University and NICTA, Tech. Rep., 2004.

[22] J. Webster and R. Watson, "Analyzing the past to prepare for the future: Writing a literature review," *MIS Quarterly*, vol. 26, no. 2, pp. xiii–xxiii, 2002. [Online]. Available: http://www.misq.org/archivist/vol/no26/issue2/GuestEd.pdf

[23] D. Benavides, S. Segura, and A. Ruiz-Cortés, "Automated analysis of feature models 20 years later: A literature review," *Information Systems*, vol. 35, no. 6, pp. 615 – 636, 2010.

[24] Y. J. M. Harman and Y. Zhang, "Achievements, open problems and challenges for search based software testing," in *8th IEEE International Conference on Software Testing, Verification and Validation*, Graz, Austria, April 2015, keynote.

[25] Y. Jia and M. Harman, "An analysis and survey of the development of mutation testing," *IEEE Trans. Softw. Eng.*, vol. 37, no. 5, pp. 649–678, Sep. 2011. [Online]. Available: http://dx.doi.org/10.1109/TSE.2010.62

[26] A. Harzing, "Publish or Perish. http://www.harzing.com/pop.htm," 2007.

[27] T. Y. Chen, D. H. Huang, T. H. Tse, and Z. Q. Zhou, "Case studies on the selection of useful relations in metamorphic testing," in *Proceedings of the 4th Ibero-American Symposium on Software Engineering and Knowledge Engineering (JIISIC 2004)*, 2004, pp. 569–583.

[28] T. Y. Chen, F.-C. Kuo, Y. Liu, and A. Tang, "Metamorphic testing and testing with special values," in *5th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, 2004, pp. 128–134.

[29] Z. Zhang, W. K. Chan, T. H. Tse, and P. Hu, "Experimental study to compare the use of metamorphic testing and assertion checking," *Journal of Software*, vol. 20, no. 10, pp. 2637–2654, 2009.

[30] T. Y. Chen, T. H. Tse, and Z. Q. Zhou, "Semi-proving: An integrated method for program proving, testing, and debugging," *IEEE Transactions on Software Engineering*, vol. 37, no. 1, pp. 109–125, Jan 2011.

[31] X. Xie, J. W. K. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, "Testing and validating machine learning classifiers by metamorphic testing," *The Journal of Systems and Software*, vol. 84, no. 4, pp. 544–558, Apr. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2010.11.920

[32] A. C. Barus, T. Y. Chen, D. Grant, F.-C. Kuo, and M. F. Lau, "Testing of heuristic methods: A case study of greedy algorithm," in *Software Engineering Techniques*, ser. Lecture Notes in Computer Science, Z. Huzar, R. Koci, B. Meyer, B. Walter, and J. Zendulka, Eds. Springer Berlin Heidelberg, 2011, vol. 4980, pp. 246–260. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-22386-0_19

[33] F.-C. Kuo, S. Liu, and T. Y. Chen, "Testing a binary space partitioning algorithm with metamorphic testing," in *Proceedings of the 2011 ACM Symposium on Applied Computing*, ser. SAC '11. New York, NY, USA: ACM, 2011, pp. 1482–1489. [Online]. Available: http://doi.acm.org/10.1145/1982185.1982502

[34] C. Sun, G. Wang, B. Mu, H. Liu, Z. Wang, and T. Y. Chen, "Metamorphic testing for web services: Framework and a case study," in *IEEE International Conference on Web Services (ICWS), 2011*, July 2011, pp. 283–290.

[35] Z.-W. Hui, S. Huang, H. Li, J.-H. Liu, and L.-P. Rao, "Measurable metrics for qualitative guidelines of metamorphic relation," in *Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual*, vol. 3, July 2015, pp. 417–422.

[36] P. Hu, Z. Zhang, W. K. Chan, and T. H. Tse, "An empirical comparison between direct and indirect test result checking approaches," in *Proceedings of the 3rd International Workshop on Software Quality Assurance*, ser. SOQUA '06. New York, NY, USA: ACM, 2006, pp. 6–13. [Online]. Available: http://doi.acm.org/10.1145/1188895.1188901

[37] T. Y. Chen, F.-C. Kuo, R. Merkel, and W. K. Tam, "Testing an open source suite for open queuing network modelling using metamorphic testing technique," in *14th IEEE International Conference on Engineering of Complex Computer Systems*, June 2009, pp. 23–29.

[38] J. Mayer and R. Guderlei, "An empirical study on the selection of good metamorphic relations," in *30th Annual International Computer Software and Applications Conference*, vol. 1, Sept 2006, pp. 475–484.

[39] Z. Q. Zhou, D. H. Huang, T. H. Tse, Z. Yang, H. Huang, and T. Y. Chen, "Metamorphic testing and its applications," in *Proceedings of the 8th International Symposium on Future Software Technology (ISFST 2004)*. Software Engineers Association, 2004.

[40] T. Y. Chen, J. W. K. Ho, H. Liu, and X. Xie, "An innovative approach for testing bioinformatics programs using metamorphic testing," *BioMed Central Bioinformatics Journal*, vol. 10, no. 1, p. 24, 2009. [Online]. Available: http://www.biomedcentral.com/1471-2105/10/24

[41] G. Batra and J. Sengupta, "An efficient metamorphic testing technique using genetic algorithm," in *Information Intelligence, Systems, Technology and Management*, ser. Communications in Computer and Information Science, S. Dua, S. Sahni, and D. Goyal, Eds. Springer Berlin Heidelberg, 2011, vol. 141, pp. 180–188. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-19423-8_19

[42] L. Chen, L. Cai, J. Liu, Z. Liu, S. Wei, and P. Liu, "An optimized method for generating cases of metamorphic testing," in *6th International Conference on New Trends in Information Science and Service Science and Data Mining (ISSDM), 2012*, Oct 2012, pp. 439–443.

[43] J. Ding, T. Wu, J. Q. Lu, and X. Hu, "Self-checked metamorphic testing of an image processing program," in *2010 Fourth International Conference on Secure Software Integration and Reliability Improvement (SSIRI)*, June 2010, pp. 190–197.

[44] G. Dong, T. Guo, and P. Zhang, "Security assurance with program path analysis and metamorphic testing," in *4th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2013*, May 2013, pp. 193–197.

[45] F.-C. Kuo, Z. Q. Zhou, J. Ma, and G. Zhang, "Metamorphic testing of decision support systems: a case study," *Software, IET*, vol. 4, no. 4, pp. 294–301, August 2010.

[46] M. Asrafi, H. Liu, and F.-C. Kuo, "On testing effectiveness of metamorphic relations: A case study," in *Fifth International Conference on Secure Software Integration and Reliability Improvement (SSIRI), 2011*, June 2011, pp. 147–156.

[47] Y. Cao, Z. Q. Zhou, and T. Y. Chen, "On the correlation between the effectiveness of metamorphic relations and dissimilarities of test case executions," in *13th International Conference on Quality Software (QSIC), 2013*, July 2013, pp. 153–162.

[48] Z. Q. Zhou, "Using coverage information to guide test case selection in adaptive random testing," in *Computer Software and Applications Conference Workshops*, July 2010, pp. 208–213.

[49] R. Just and F. Schweiggert, "Automating software tests with partial oracles in integrated environments," in *Proceedings of the 5th Workshop on Automation of Software Test*, ser. AST '10. New York, NY, USA: ACM, 2010, pp. 91–94. [Online]. Available: http://doi.acm.org/10.1145/1808266.1808280

[50] ——, "Automating unit and integration testing with partial oracles," *Software Quality Journal*, vol. 19, no. 4, pp. 753–769, 2011. [Online]. Available: http://dx.doi.org/10.1007/s11219-011-9151-x

[51] X. Xie, J. Tu, T. Y. Chen, and B. Xu, "Bottom-up integration testing with the technique of metamorphic testing," in *14th International Conference on Quality Software (QSIC), 2014*, Oct 2014, pp. 73–78.

[52] W. K. Chan, T. Y. Chen, H. Lu, T. H. Tse, and S. S. Yau, "Integration testing of context-sensitive middleware-based applications: a metamorphic approach." *International Journal of Software Engineering and Knowledge Engineering*, vol. 16, no. 5, pp. 677–704, 2006. [Online]. Available: http://dblp.uni-trier.de/db/journals/ijseke/ijseke16.html#ChanCLTY06

[53] Z. Hui and S. Huang, "A formal model for metamorphic relation decomposition," in *Fourth World Congress on Software Engineering (WCSE), 2013*, Dec 2013, pp. 64–68.

[54] H. Liu, X. Liu, and T. Y. Chen, "A new method for constructing metamorphic relations," in *12th International Conference on Quality Software (QSIC), 2012*, Aug 2012, pp. 59–68.

[55] G. Dong, B. Xu, L. Chen, C. Nie, and L. Wang, "Case studies on testing with compositional metamorphic relations," *Journal of Southeast University (English Edition)*, vol. 24, no. 4, pp. 437–443, 2008.

[56] U. Kanewala and J. M. Bieman, "Using machine learning techniques to detect metamorphic relations for programs without test oracles," in *IEEE 24th International Symposium on Software Reliability Engineering (ISSRE), 2013*, Nov 2013, pp. 1–10.

[57] U. Kanewala, "Techniques for automatic detection of metamorphic relations," in *IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2014*, March 2014, pp. 237–238.

[58] C. Murphy, G. Kaiser, and L. Hu, "Properties of machine learning applications for use in metamorphic testing," Department of Computer Science, Columbia University, New York NY, Tech. Rep., 2008.

[59] U. Kanewala, J. M. Bieman, and A. Ben-Hur, "Predicting metamorphic relations for testing scientific software: a machine learning approach using graph kernels," *Software Testing, Verification and Reliability*, 2015. [Online]. Available: http://dx.doi.org/10.1002/stvr.1594

[60] J. Zhang, J. Chen, D. Hao, Y. Xiong, B. Xie, L. Zhang, and H. Mei, "Search-based inference of polynomial metamorphic relations," in *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, ser. ASE '14. New York, NY, USA: ACM, 2014, pp. 701–712. [Online]. Available: http://doi.acm.org/10.1145/2642937.2642994

[61] A. Carzaniga, A. Goffi, A. Gorla, A. Mattavelli, and M. Pezzè, "Cross-checking oracles from intrinsic software redundancy," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 931–942. [Online]. Available: http://doi.acm.org/10.1145/2568225.2568287

[62] A. Goffi, A. Gorla, A. Mattavelli, M. Pezze, and P. Tonella, "Search-based synthesis of equivalent method sequences," in *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE 2014. New York, NY, USA: ACM, 2014, pp. 366–376. [Online]. Available: http://doi.acm.org/10.1145/2635868.2635888

[63] A. Goffi, "Automatic generation of cost-effective test oracles," in *Companion Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE Companion 2014. New York, NY, USA: ACM, 2014, pp. 678–681. [Online]. Available: http://doi.acm.org/10.1145/2591062.2591078

[64] F. Su, J. Bell, C. Murphy, and G. Kaiser, "Dynamic inference of likely metamorphic properties to support differential testing," in *Proceedings of the 10th International Workshop on Automation of Software Test*, ser. AST '15. Piscataway, NJ, USA: IEEE Press, 2015, pp. 55–59. [Online]. Available: http://dl.acm.org/citation.cfm?id=2819261.2819279

[65] T. Y. Chen, P. Poon, and X. Xie, "METRIC: METamorphic Relation Identification based on the Category-choice framework," *Journal of Systems and Software*, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121215001624

[66] T. Y. Chen, P.-L. Poon, S.-F. Tang, and T. H. Tse, "Dessert: a divide-and-conquer methodology for identifying categories, choices, and choice relations for test case generation," *Software Engineering, IEEE Transactions on*, vol. 38, no. 4, pp. 794–809, July 2012.

[67] A. Gotlieb and B. Botella, "Automated metamorphic testing," in *Proceedings of the 27th Annual International Conference on Computer Software and Applications*, ser. COMPSAC '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 34–. [Online]. Available: http://dl.acm.org/citation.cfm?id=950785.950794

[68] P. Wu, X. Shi, J. Tang, H. Lin, and T. Y. Chen, "Metamorphic testing and special case testing: A case study," *Journal of Software*, vol. 16, pp. 1210–1220, 2005.

[69] S. Segura, R. M. Hierons, D. Benavides, and A. Ruiz-Cortés, "Automated metamorphic testing on the analyses of feature models," *Information and Software Technology*, vol. 53, no. 3, pp. 245 – 258, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584910001904

[70] P. Wu, "Iterative metamorphic testing," in *29th Annual International Computer Software and Applications Conference, 2005. COMPSAC 2005*, vol. 1, July 2005, pp. 19–24.

[71] G. Dong, C. Nie, B. Xu, and L. Wang, "An effective iterative metamorphic testing algorithm based on program path analysis," in *eventh International Conference on Quality Software, 2007. QSIC '07*, Oct 2007, pp. 292–297.

[72] S. Segura, R. M. Hierons, D. Benavides, and A. Ruiz-Cortes, "Automated test data generation on the analyses of feature models: A metamorphic testing approach," in *Third International Conference on Software Testing, Verification and Validation (ICST), 2010*, April 2010, pp. 35–44.

[73] S. Segura, A. Durán, A. B. Sánchez, D. L. Berre, E. Lonca, and A. Ruiz-Cortés, "Automated metamorphic testing of variability analysis tools," *Software Testing, Verification and Reliability*, vol. 25, no. 2, pp. 138–163, 2015. [Online]. Available: http://dx.doi.org/10.1002/stvr.1566

[74] R. Guderlei and J. Mayer, "Statistical metamorphic testing: Testing programs with random output by means of statistical hypothesis tests and metamorphic testing," in *Seventh International Conference on Quality Software, 2007. QSIC '07*, Oct 2007, pp. 404–409.

[75] C. Murphy, M. S. Raunak, A. King, S. Chen, C. Imbriano, G. Kaiser, I. Lee, O. Sokolsky, L. Clarke, and L. Osterweil, "On effective testing of health care simulation software," in *Proceedings of the 3rd Workshop on Software Engineering in Health Care*, ser. SEHC '11. New York, NY, USA: ACM, 2011, pp. 40–47. [Online]. Available: http://doi.acm.org/10.1145/1987993.1988003

[76] C. Murphy, "Using runtime testing to detect defects in applications without test oracles," in *Proceedings of the 2008 Foundations of Software Engineering Doctoral Symposium*, ser. FSEDS '08. New York, NY, USA: ACM, 2008, pp. 21–24. [Online]. Available: http://doi.acm.org/10.1145/1496653.1496659

[77] C. Murphy, K. Shen, and G. Kaiser, "Using jml runtime assertion checking to automate metamorphic testing in applications without test oracles," in *Second International Conference on Software Testing Verification and Validation, ICST 2009*, 2009.

[78] "Java Modeling Language (JML). http://www.eecs.ucf.edu/~leavens/JML//index.shtml," accessed on May 2015.

[79] C. Murphy, G. Kaiser, L. Hu, and L. Wu, "Properties of machine learning applications for use in metamorphic testing." in *International conference on software engineering and knowledge engineering*, 2008, pp. 867—-872.

[80] C. Murphy, K. Shen, and G. Kaiser, "Automatic system testing of programs without test oracles," in *Proceedings of the Eighteenth International Symposium on Software Testing and Analysis*, ser. ISSTA '09. New York, NY, USA: ACM, 2009, pp. 189–200. [Online]. Available: http://doi.acm.org/10.1145/1572272.1572295

[81] H. Zhu, "Jfuzz: A tool for automated java unit testing based on data mutation and metamorphic testing methods," in *Trustworthy Systems and Their Applications (TSA), 2015 Second International Conference on*, July 2015, pp. 8–15.

[82] W. K. Chan, S. C. Cheung, and K. R. P. Leung, "Towards a metamorphic testing methodology for service-oriented software applications," in *Fifth International Conference on Quality Software, 2005. (QSIC 2005)*, Sept 2005, pp. 470–476.

[83] W. K. Chan, S. C. Cheung, and K. R. P. H. Leung, "A metamorphic testing approach for online testing of service-oriented software applications." *International Journal of Web Services Research*, vol. 4, no. 2, pp. 61–81, 2007. [Online]. Available: http://dblp.uni-trier.de/db/journals/jwsr/jwsr4.html#ChanCL07

[84] C. Sun, G. Wang, B. Mu, H. Liu, Z. Wang, and T. Y. Chen, "A metamorphic relation-based approach to testing web services without oracles," *International Journal of Web Services Research*, vol. 9, no. 1, pp. 51–73, Jan. 2012. [Online]. Available: http://dx.doi.org/10.4018/jwsr.2012010103

[85] C. Castro-Cabrera and I. Medina-Bulo, "An approach to metamorphic testing for ws-bpel compositions," in *Proceedings of the International Conference on e-Business (ICE-B), 2011*, July 2011, pp. 1–6.

[86] ——, "Application of metamorphic testing to a case study in web services compositions," in *E-Business and Telecommunications*, ser. Communications in Computer and Information Science, M. Obaidat, J. Sevillano, and J. Filipe, Eds. Springer Berlin Heidelberg, 2012, vol. 314, pp. 168–181. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35755-8_13

[87] "OASIS: Web Services Business Process Execution Language 2.0. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS. html," accessed on May 2015.

[88] Z. Q. Zhou, T. H. Tse, F.-C. Kuo, and T. Y. Chen, "Automated functional testing of web search engines in the absence of an oracle," Department of Computer Science, The University of Hong Kong, Tech. Rep. TR-2007-06, 2007.

[89] Z. Q. Zhou, S. Xiang, and T. Y. Chen, "Metamorphic testing for software quality assessment: A study of search engines," *Software Engineering, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

[90] J. Mayer and R. Guderlei, "On random testing of image processing applications," in *Sixth International Conference on Quality Software, 2006. QSIC 2006*, Oct 2006, pp. 85–92.

[91] R. Guderlei and J. Mayer, "Towards automatic testing of imaging software by means of random and metamorphic testing," *International Journal of Software Engineering and Knowledge Engineering*, vol. 17, no. 06, pp. 757–781, 2007. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S0218194007003471

[92] W. K. Chan, J. C. F. Ho, and T. H. Tse, "Piping classification to metamorphic testing: An empirical study towards better effectiveness for the identification of failures in mesh simplification programs," in *31st Annual International Computer Software and Applications Conference, 2007. COMPSAC 2007*, vol. 1, July 2007, pp. 397–404.

[93] ——, "Finding failures from passed test cases: Improving the pattern classification approach to the testing of mesh simplification programs," *Software Testing, Verification and Reliability Journal*, vol. 20, no. 2, pp. 89–120, Jun. 2010. [Online]. Available: http://dx.doi.org/10.1002/stvr.v20:2

[94] R. Just and F. Schweiggert, "Evaluating testing strategies for imaging software by means of mutation analysis," in *International Conference on Software Testing, Verification and Validation Workshops, 2009. ICSTW '09*, April 2009, pp. 205–209.

[95] T. Jameel, L. Mengxiang, and C. Liu, "Test oracles based on metamorphic relations for image processing applications," in *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2015 16th IEEE/ACIS International Conference on*, June 2015, pp. 1–6.

[96] T. H. Tse, S. S. Yau, W. K. Chan, H. Lu, and T. Y. Chen, "Testing context-sensitive middleware-based software applications," in *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, Sept 2004, pp. 458–466 vol.1.

[97] W. K. Chan, T. Y. Chen, H. Lu, T. H. Tse, and S. S. Yau, "A metamorphic approach to integration testing of context-sensitive middleware-based applications," in *Fifth International Conference on Quality Software, 2005. (QSIC 2005)*, Sept 2005, pp. 241–249.

[98] W. K. Chan, T. Y. Chen, S. C. Cheung, T. H. Tse, and Z. Zhang, "Towards the testing of power-aware software applications for wireless sensor networks," in *Ada Europe 2007 - Reliable Software Technologies*, ser. Lecture Notes in Computer Science, N. Abdennadher and F. Kordon, Eds. Springer Berlin Heidelberg, 2007, vol. 4498, pp. 84–99. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-73230-3_7

[99] F.-C. Kuo, T. Y. Chen, and W. K. Tam, "Testing embedded software by metamorphic testing: A wireless metering system case study," in *IEEE 36th Conference on Local Computer Networks (LCN), 2011*, Oct 2011, pp. 291–294.

[100] M. Jiang, T. Y. Chen, F.-C. Kuo, and Z. Ding, "Testing central processing unit scheduling algorithms using metamorphic testing," in *4th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2013*, May 2013, pp. 530–536.

[101] K. Y. Sim, W. K. S. Pao, and C. Lin, "Metamorphic testing using geometric interrogation technique and its application," in *Proceedings of the 2nd International Conference of Electrical Engineering/Electronics, Computer, Telecommunications, and Information Technology*, 2005, pp. 91–95.

[102] T. Y. Chen, F.-C. Kuo, H. Liu, and S. Wang, "Conformance testing of network simulators based on metamorphic testing technique," in *Formal Techniques for Distributed Systems*, ser. Lecture Notes in Computer Science, D. Lee, A. Lopes, and A. Poetzsch-Heffter, Eds. Springer Berlin Heidelberg, 2009, vol. 5522, pp. 243–248. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02138-1_19

[103] "OMNeT++ system. http://www.omnetpp.org," accessed on April 2015.

[104] J. Ding, T. Wu, D. Xu, J. Q. Lu, and X. Hu, "Metamorphic testing of a monte carlo modeling program," in *Proceedings of the 6th International Workshop on Automation of Software Test*, ser. AST '11. New York, NY, USA: ACM, 2011, pp. 1–7. [Online]. Available: http://doi.acm.org/10.1145/1982595.1982597

[105] A. Nuñez and R. M. Hierons, "A methodology for validating cloud models using metamorphic testing," *annals of telecommunications - annales des télécommunications*, pp. 1–9, 2014. [Online]. Available: http://dx.doi.org/10.1007/s12243-014-0442-7

[106] A. Nuñez, J. L. Vazquez-Poletti, A. C. Caminero, G. G. Castañe, J. Carretero, and I. M. Llorente, "icancloud: A flexible and scalable cloud infrastructure simulator," *Journal of Grid Computing*, vol. 10, no. 1, pp. 185–209, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10723-012-9208-5

[107] P. C. Cañizares, A. Núñez, M. Núñez, and J. J. Pardo, "A methodology for designing energy-aware systems for computational science," *Procedia Computer Science*, vol. 51, pp. 2804 – 2808, 2015, international Conference On Computational Science, {ICCS} 2015Computational Science at the Gates of Nature. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050915012466

[108] X. Xie, J. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, "Application of metamorphic testing to supervised classifiers," in *9th International Conference on Quality Software, 2009. QSIC '09.*, Aug 2009, pp. 135–144.

[109] J. E. Gewehr, M. Szugat, and R. Zimmer, "Bioweka—extending the weka framework for bioinformatics," *Bioinformatics*, vol. 23, no. 5, pp. 651–653, Feb. 2007. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btl671

[110] Z. Jing, H. Xuegang, and Z. Bin, "An evaluation approach for the program of association rules algorithm based on metamorphic relations," *Journal of Electronics (China)*, vol. 28, no. 4-6, pp. 623–631, 2011. [Online]. Available: http://dx.doi.org/10.1007/s11767-012-0743-9

[111] L. L. Pullum and O. Ozmen, "Early results from metamorphic testing of epidemiological models," in *ASE/IEEE International Conference on BioMedical Computing (BioMedCom), 2012*, Dec 2012, pp. 62–67.

[112] A. Ramanathan, C. A. Steed, and L. L. Pullum, "Verification of compartmental epidemiological models using metamorphic testing, model checking and visual analytics," in *ASE/IEEE International Conference on BioMedical Computing (BioMedCom), 2012*, Dec 2012, pp. 68–73.

[113] S. Beydeda, "Self-metamorphic-testing components," in *30th Annual International Computer Software and Applications Conference, 2006. COMPSAC '06*, vol. 2, Sept 2006, pp. 265–272.

[114] X. Lu, Y. Dong, and C. Luo, "Testing of component-based software: A metamorphic testing methodology," in *International Conference on Ubiquitous Intelligence Computing and International Conference on Autonomic Trusted Computing*, Oct 2010, pp. 272–276.

[115] T. Y. Chen, J. Feng, and T. H. Tse, "Metamorphic testing of programs on partial differential equations: A case study," in *Proceedings of the 26th International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment*, ser. COMPSAC '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 327–333. [Online]. Available: http://dl.acm.org/citation.cfm?id=645984.675903

[116] C. Aruna and R. S. R. Prasad, "Metamorphic relations to improve the test accuracy of multi precision arithmetic software applications," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI, 2014*, Sept 2014, pp. 2244–2248.

[117] Q. Tao, W. Wu, C. Zhao, and W. Shen, "An automatic testing approach for compiler based on metamorphic testing technique," in *17th Asia Pacific Software Engineering Conference (APSEC), 2010*, Nov 2010, pp. 270–279.

[118] V. Le, M. Afshari, and Z. Su, "Compiler validation via equivalence modulo inputs," in *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI '14. New York, NY, USA: ACM, 2014, pp. 216–226. [Online]. Available: http://doi.acm.org/10.1145/2594291.2594334

[119] T. Y. Chen, F.-C. Kuo, and Z. Q. Zhou, "An effective testing method for end-user programmers," in *Proceedings of the First Workshop on End-user Software Engineering*, ser. WEUSE I. New York, NY, USA: ACM, 2005, pp. 1–5. [Online]. Available: http://doi.acm.org/10.1145/1082983.1083236

[120] K. Y. Sim, C. S. Low, and F.-C. Kuo, "Detecting faults in technical indicator computations for financial market analysis," in *2nd International Conference on Information Science and Engineering (ICISE), 2010*, Dec 2010, pp. 2749–2754.

[121] "MetaTrader 4 Trading Terminal. http://www.metaquotes.net/en/metatrader4/trading_terminal," accessed April 2015.

[122] S. Yoo, "Metamorphic testing of stochastic optimisation," in *Third International Conference on Software Testing, Verification, and Validation Workshops (ICSTW), 2010*, April 2010, pp. 192–201.

[123] Y. Yao, S. Huang, and M. Ji, "Research on metamorphic testing for oracle problem of integer bugs," in *Fourth International Conference on Advances in Computer Science and Information Engineering*, ser. Advances in Intelligent and Soft Computing, D. Jin and S. Lin, Eds. Springer Berlin Heidelberg, 2012, vol. 168, pp. 95–100. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-30126-1_16

[124] Y. Yao, C. Zheng, S. Huang, and Z. Ren, "Research on metamorphic testing: A case study in integer bugs detection," in *Fourth International Conference on Intelligent Systems Design and Engineering Applications, 2013*, Nov 2013, pp. 488–493.

[125] Z. Hui, S. Huang, Z. Ren, and Y. Yao, "Metamorphic testing integer overflow faults of mission critical program: A case study," *Mathematical Problems in Engineering*, vol. 2013, 2013.

[126] G. Batra and G. Singh, "An automated metamorphic testing technique for designing effective metamorphic relations," in *Contemporary Computing*, ser. Communications in Computer and Information Science, M. Parashar, D. Kaushik, O. Rana, R. Samtaney, Y. Yang, and A. Zomaya, Eds. Springer Berlin Heidelberg, 2012, vol. 306, pp. 152–163. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-32129-0_20

[127] C. Sun, Z. Wang, and G. Wang, "A property-based testing framework for encryption programs," *Frontiers of Computer Science*, vol. 8, no. 3, pp. 478–489, 2014. [Online]. Available: http://dx.doi.org/10.1007/s11704-014-3040-y

[128] C. Aruna and R. S. R. Prasad, "Adopting metamorphic relations to verify non-testable graph theory algorithms," in *Advances in Computing and Communication Engineering (ICACCE), 2015 Second International Conference on*, May 2015, pp. 673–678.

[129] M. Lindvall, D. Ganesan, R. Ardal, and R. Wiegand, "Metamorphic model-based testing applied on nasa dat – an experience report," in *Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on*, vol. 2, May 2015, pp. 129–138.

[130] T. Y. Chen, T. H. Tse, and Z. Q. Zhou, "Fault-based testing in the absence of an oracle," in *Proceedings of the 25th Annual International Computer Software and Applications Conference (COMPSAC 2001)*. IEEE Computer Society Press, 2001, pp. 172–178.

[131] C. Cadar and K. Sen, "Symbolic execution for software testing: Three decades later," *Communications of the ACM*, vol. 56, no. 2, pp. 82–90, Feb. 2013. [Online]. Available: http://doi.acm.org/10.1145/2408776.2408795

[132] I. Erete and A. Orso, "Optimizing constraint solving to better support symbolic execution," in *Workshop on Constraints in Software Testing, Verification, and Analysis*, 2011.

[133] T. Y. Chen, T. H. Tse, and Z. Q. Zhou, "Semi-proving: An integrated method based on global symbolic evaluation and metamorphic testing," in *Proceedings of the 2002 ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA '02. New York, NY, USA: ACM, 2002, pp. 191–195. [Online]. Available: http://doi.acm.org/10.1145/566172.566202

[134] G. Dong, S. Wu, G. Wang, T. Guo, and Y. Huang, "Security assurance with metamorphic testing and genetic algorithm," in *IEEE/WIC/ACM International Conference onWeb Intelligence and Intelligent Agent Technology (WI-IAT), 2010*, vol. 3, Aug 2010, pp. 397–401.

[135] X. Xie, W. E. Wong, T. Y. Chen, and B. Xu, "Spectrum-based fault localization: Testing oracles are no longer mandatory," in *11th International Conference on Quality Software (QSIC), 2011*, July 2011, pp. 1–10.

[136] ——, "Metamorphic slice: An application in spectrum-based fault localization," *Information and Software Technology*, vol. 55, no. 5, pp. 866 – 879, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584912001759

[137] "Siemens Suite. http://sir.unl.edu/portal/bios/tcas.php#siemens," accessed May 2015.

[138] Y. Lei, X. Mao, and T. Y. Chen, "Backward-slice-based statistical fault localization without test oracles," in *13th International Conference on Quality Software (QSIC), 2013*, July 2013, pp. 212–221.

[139] Y. Lei, X. Mao, Z. Dai, and C. Wang, "Effective statistical fault localization using program slices," in *Computer Software and Applications Conference*, July 2012, pp. 1–10.

[140] P. Rao, Z. Zheng, T. Y. Chen, N. Wang, and K. Cai, "Impacts of test suite's class imbalance on spectrum-based fault localization techniques," in *13th International Conference on Quality Software (QSIC), 2013*, July 2013, pp. 260–267.

[141] C. Aruna and R. S. R. Prasad, "Testing approach for dynamic web applications based on automated test strategies," in *ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India- Vol II*, ser. Advances in Intelligent Systems and Computing, S. C. Satapathy, P. S. Avadhani, S. K. Udgata, and S. Lakshminarayana, Eds. Springer International Publishing, 2014, vol. 249, pp. 399–410. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-03095-1_43

[142] S. Artzi, A. Kiezun, J. Dolby, F. Tip, D. Dig, A. Paradkar, and M. D. Ernst, "Finding bugs in dynamic web applications," in *International Symposium on Software Testing and Analysis*, ser. ISSTA '08. New York, NY, USA: ACM, 2008, pp. 261–272. [Online]. Available: http://doi.acm.org/10.1145/1390630.1390662

[143] H. Liu, I. I. Yusuf, H. W. Schmidt, and T. Y. Chen, "Metamorphic fault tolerance: An automated and systematic methodology for fault tolerance in the absence of test oracle," in *Companion Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE Companion 2014. New York, NY, USA: ACM, 2014, pp. 420–423. [Online]. Available: http://doi.acm.org/10.1145/2591062.2591109

[144] H. Jin, Y. Jiang, N. Liu, C. Xu, X. Ma, and J. Lu, "Concolic metamorphic debugging," in *Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual*, vol. 2, July 2015, pp. 232–241.

[145] H. Liu, F.-C. Kuo, and T. Y. Chen, "Teaching an end-user testing methodology," in *23rd IEEE Conference on Software Engineering Education and Training (CSEE T)*, March 2010, pp. 81–88.

[146] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: a survey," *Software Testing, Verification and Reliability*, vol. 22, no. 2, pp. 67–120, 2012. [Online]. Available: http://dx.doi.org/10.1002/stvr.430

[147] M. D. Ernst, J. Cockrell, W. G. Griswold, and D. Notkin, "Dynamically discovering likely program invariants to support program evolution," in *Proceedings of the 21st International Conference on Software Engineering*, ser. ICSE '99. New York, NY, USA: ACM, 1999, pp. 213–224. [Online]. Available: http://doi.acm.org/10.1145/302405.302467

APPENDIX A
SUBJECT PROGRAMS IN METAMORPHIC TESTING

Name	Language	Size	Description	References
TCAS	C/C++	173	Onboard aircraft conflict detection (Siemens suite)	[35], [46], [123], [125], [135], [136], [138], [140]
Weka	Java	NR	Machine learning application	[31], [51], [64], [77], [80], [108], [110]
Grep	C/C++	1006	Command-line tool for pattern matching	[47], [135], [136], [138], [140]
Replace	C/C++	563	Regular expression matching and substitutions (Siemens suite)	[30], [135], [136], [138], [140]
Print_tokens	C/C++	342	Lexical analyzer (Siemens suite)	[135], [136], [138], [140]
Print_tokens2	C/C++	355	Lexical analyzer (Siemens suite)	[135], [136], [138], [140]
Schedule	C/C++	292	Priority scheduler (Siemens suite)	[135], [136], [138], [140]
Schedule2	C/C++	262	Priority scheduler (Siemens suite)	[135], [136], [138], [140]
Tot_info	C/C++	273	Information measure (Siemens suite)	[135], [136], [138], [140]
TriSquareJ	Java	30	Returns the type and square of a triangle	[42], [44], [71], [134]
Boyer	Java	241	Returns the index of the first occurrence of a pattern within a text	[29], [36], [144]
FaMa	Java	NR	Feature model analysis tool	[69], [72], [73]
GCC	C/C++	NR	C compiler	[117], [118]
GeoStoch	Java	NR	Matrix determinant computation	[38], [90], [91]
Google	Online	N/A	Online search engine	[20], [88], [89]
JJ2000 library	Java	NR	jpeg2000 image encoder/decoder	[49], [50], [94]
PAYL	C/C++	NR	Anomaly-based instrusion detection system	[58], [80], [116]
SeqMap	C/C++	1783	Tool for mapping short sequence reads to a reference genome	[40], [135], [136]
ShortestPath	Java	271	Mesh simplification algorithm	[27], [92], [128]
SpMatMult	C/C++	35	Sparse matrix multiplication (JASPA benchmark)	[55], [68], [70]
Trityp	C/C++	30	Triangle classification program	[41], [55], [67]
CommonsMath1	Java	1626	Apache Mathematical library	[59], [60]
ATM	Java	136	Automatic teller machine web service	[34], [84]
Jboolexpr	Java	231	Boolean string expressions evaluation	[29], [36]
Bsearch	C/C++	43	Binary search within a sorted array	[28], [67]
Knapsack	Java	180	Knapsack optimization algorithm	[46], [64]
LiveSearch	Online	N/A	Online search engine	[20], [88]
MartiRank	NR	NR	Ranking algorithm	[58], [80]
Melax	Java	NR	Polygon reduction algorithm	[92], [93]
Quadric	Java	NR	Mesh simplification algorithm	[92], [93]
QuadricTri	Java	NR	Mesh simplification algorithm	[92], [93]
Sine	C/C++	99	Sine calculation	[28], [35]
SPLAR	Java	NR	Feature model analysis tool	[69], [73]
TxnTableSorter	Java	281	Personal accounting software	[29], [36]
Yahoo	Online	N/A	Online search engine	[20], [88]
2D-MatrixSearch	Java	34	Searches for a value in an m x n matrix	[144]
35Math	Java	7–45	35 mathematical functions	[56]
3DCell	Fortran90	5600	3D cell structure reconstruction	[43]
Apache Mahout	Java	NR	Machine learning library	[59]
Arhant-II	C/C++	NR	Real-time mathematical C project	[116]
Aspcudf	C/C++		CUDF document analyser	[73]
Baidu	NR	NR	Online search engine	[89]
Bank	C#	NR	Banking system application	[126]
Bash	C/C++	5984	Command-line interpreter	[47]
BigInt	C/C++	500	Calculator for very large integers	[47]
Bing	NR	NR	Online search engine	[89]
BSP-TreeVS	C/C++	NR	Surface visibility using Binary Space Partitioning (BSP) tree	[33]
C4.5	C/C++	NR	Algorithm for building decision trees	[80]
Cabot	Java	NR	LANDMARC RFID-based location sensing algorithm	[52]
Chinese Bing	NR	NR	Online search engine	[89]
Clasp	NR		Conflict-driven answer SAT solver	[73]
ClosestPair	Java	370	Princeton algorithm for finding the closest pair	[144]
CommonsMath2	Java	NR	Matrix determinant computation	[38]
ConnectedC8	Java	215	Labeling of connected components in binary images	[91]
CpWiki	C/C++	125	Return the longest path in a graph and its length	[47]
CriticalPath	NR	NR	Return the longest path in a graph and its length	[27]
Cudf-check	C/C++		CUDF document analyser	[73]
Decider	Java	12795	Decision support system	[45]
DecodingWays	Java	78	Return the number of ways to decode an encoded message	[144]
Determinant1	Java	NR	Matrix determinant computation (Michael Flanagan's implementation)	[38]
Determinant2	Java	NR	Matrix determinant computation (Jon Squire's implementation)	[38]
Determinant3	Java	30	Matrix determinant computation	[134]
DistinctSubsequence	Java	32	Count the distinct subsequences of an string in another string	[144]
Dnapars	NR	NR	Phylogenetic program	[54]
Editing distance	Java	73	Enhanced edit distance algorithm	[144]
Edmonds-Karp	Java	229	Maximum flow algorithm	[144]
FindKNN	Java	153	Finding the k nearest neighbors of a sample point	[4]
FirstMissingPositive	Java	40	Find the first missing positive integer in an unsorted integer array	[144]
FLAME	Prolog		Feature model analysis tool	[73]

continued on next page

Name	Language	Size	Description	References
GBT	C/C++	NR	Real-time mathematical C project	[116]
GCS	MATLAB	NR	Loop insulin titration simulator	[75]
GetMid	C/C++	17	Compute the median of three integers	[67]
GNLab	C/C++	NR	Analysis and simulation of gene regulatory networks	[40]
Grade	C/C++	2035	Grade computation module	[35]
GraphStream	Java	NR	Modeling and analysis of graphs	[61]
Guava	Java	NR	Google utility library	[61]
HeapSort	Java	66	Heap sort algorithm	[144]
HillCipher	C/C++	74	Hill cipher encryption program	[127]
HRRN1	NR	NR	Highest Response Ratio Next (HRRN) scheduler simulator	[100]
HRRN2	NR	NR	Highest Response Ratio Next (HRRN) scheduler simulator	[100]
ImageDilation	C/C++	NR	Binary image dilation	[95]
InterleavingString	Java	73	Find whether an string is formed by the interleaving of other two strings	[144]
InvCum	Java	90	Inverse cumulative distribution function of the normal distribution	[74]
JAMA	Java	NR	Linear algebra package	[38]
JMT	Java	NR	Calculate the major outputs of the queuing network systems	[37]
Joda-Time	Java	NR	Date and time utilities	[61]
Jscience	Java	NR	Scientific calculations and visualizations	[38]
Jsim	Java	NR	Discrete even simulator	[75]
Kcnfs	C/C++		SAT Solver	[73]
KLP	Java	36	Key-lock problem algorithm	[32]
LargestRectangle	Java	77	Find the area of the largest rectangle in a histogram	[144]
Lingeling	C/C++		SAT Solver	[73]
Lipschitz	Java	320	Computation of the Lipschitz cover	[91]
LLVM	C/C++	NR	C compiler	[118]
March_ks	C/C++		SAT Solver	[73]
March_rw	C/C++		SAT Solver	[73]
MaxRectangle	Java	113	Find the largest rectangle in a 2D binary matrix	[144]
MaxSUB	Java	25	Kadane's MAXSUB algorithm	[144]
MaxTreePathSum	Java	74	Given a binary tree, find the maximum path sum	[144]
MetaTrader	C/C++	NR	Online trading software platform	[120]
MinimizeDFA	Java	929	Minimize a deterministic finite automation	[4]
MinInRot	Java	34	Find the minimum element in a sorted and rotated array	[144]
Minisat	C/C++		SAT Solver	[73]
MinSpanTree	NR	NR	Dijkstra's algorithm to find the minimal spanning tree	[128]
MonteCarlo	Fortran91	1600	Monte Carlo modelling program	[104]
Multi-MAXSUM	Java	61	Multi-segment MAXSUM algorithm	[144]
MultipleKnapscack	Java	808	Solve the multiple knapsack problem	[4]
NASADAT	NR	NR	NASA database of telemetry data and query interface	[129]
NormDist	NR	36	Normal distribution probability computation	[44]
OMNeT++	C/C++	NR	Network simulator	[102]
P2cudf	Java		CUDF document analyser	[73]
PartialDiff	NR	NR	Partial differential equation calculation	[115]
PCC	C/C++	NR	C compiler	[115]
Picosat	C/C++		SAT Solver	[73]
Prim	Java	765	Compute a minimum spanning forest using Prim's MST algorithm	[144]
QuickSort	Java	49	Quick sort algorithm	[144]
RapidMiner	Java	NR	Analytic platform application	[77]
RF-Soft	C/C++	NR	Wireless metering program	[99]
RMB converter service	NR	NR	Currency converter web service	[84]
RSA	C/C++	28	RSA encryption program	[127]
Rsat	C/C++		SAT solver	[73]
Sat4j	Java	NR	SAT Solver	[73]
SCAR	NR	NR	Company car and expense claim system	[65]
SearchInRot	Java	53	Find a target value in a sorted rotated array	[144]
Sed	C/C++	1442	Stream editor that perform text transformations on an input stream	[47]
Seismic web service	Java	551	Seismic data query web service	[84]
Servcalc	C/C++	2480	Service-oriented calculator	[83]
SetCover	Java	211	Solve the set coverage problem using a greedy algorithm	[4]
Shortest	Java	NR	Mesh simplification algorithm	[92]
SimAnnealing	Java	25	Simulated annealing search	[122]
SMOS	NR	NR	Meal ordering system	[65]
SparseMatrixMultiply	Java	259	Mutiply two sparse matrices	[4]
SpStudent	C/C++	200	Find the shortest and the second shortest path between two vertices in a graph	[47]
SpWiki	C/C++	95	Shortest path between between two vertices in a graph	[47]
Superstring	NR	NR	Find the shortest common string	[64]
SurroundedRegion	Java	78	Given a board containing 'X' and 'O', capture all regions surrounded by 'X'	[144]
SVM	C/C++	NR	Real-time mathematical C project	[116]
SVM-Light	C/C++	NR	Vector Machine learning application	[58]
TCC	C/C++	NR	C compiler	[117]
Colt project	Java	NR	Scientific and technical computing library	[59]
Triangle	NR	12	Calculate triangle area (Heron's formula)	[124]
TrisquareC	C/C++	168	Calculate triangle area	[35]
UCC	C/C++	NR	C compiler	[117]

APPENDIX B
DATA EXTRACTION FORMS

B.1 List of surveyed papers

1) Chen et al. TR'98
2) Chen et al. COMPSAC'01
3) Chen et al. COMPSAC'02
4) Chen et al. ISSTA'02
5) Chen et al. IST'03
6) Gotlieb and Botella COMPSAC'03
7) Chen et al. IBCSE'04
8) Chen et al. SNPD'04
9) Chen et al. STEP'04
10) Tse et al. COMPSAC'04
11) Zhou et al. ISFST'04
12) Chan et al. QSIC'05
13) Chan et al. QSIC'05 (b)
14) Chen et al. WEUSE'05
15) Sim et al. EEEC'05
16) Tse COMPSAC'05
17) Wu COMPSAC'05
18) Wu et al. JS'05
19) Beydeda COMPSAC'06
20) Chan et al. IJSEKE'06
21) Hu et al. SOQUA'06
22) Mayer and Guderlei COMPSAC'06
23) Mayer and Guderlei QSIC'06
24) Chan et al. COMPSAC'07
25) Chan et al. IJWSR'07
26) Chan et al. RST'07
27) Dong et al. QSIC'07
28) Guderlei and Mayer IJSEKE'07
29) Guderlei and Mayer QSIC'07
30) Zhou et al. TR'07
31) Dong et al. JSU'08
32) Murphy FSEDS'08
33) Murphy et al. TR'08
34) Chan et al. STVR'09
35) Chen et al. BIOINFORMATICS'09
36) Chen et al. FTDS'09
37) Chen et al. ICECCS'09
38) Just and Schweiggert ICSTW'09
39) Murphy et al. ICST'09
40) Murphy et al. ISSTA'09
41) Xie et al. QSIC'09
42) Zhang et al. JS'09
43) Chen SOSE'10
44) Chen et al TSE'10
45) Ding et al SSIRI'10
46) Dong et al ICWIIAT'10
47) Just and Schweiggert AST'10
48) Kuo et al. IET'10
49) Liu et al. CSEET'10
50) Lu et al. UATC'10
51) Segura et al. ICST'10
52) Segura et al. IST'10
53) Sim et al. ICISE'10
54) Tao et al. APSEC'10
55) Xie et al. JSS'10
56) Yoo ICSTW'10
57) Zhou et al. STVR'10
58) Asrafi et al. SSIRI'11
59) Barus et al. SET'11
60) Batra and Sengupta ISTM'11
61) Castro-Cabrera and Medina-Bulo ICEB'11
62) Ding et al. AST'11
63) Jing et al. JE'11
64) Just and Schweiggert SQJ'11
65) Kuo et al. LCN'11
66) Kuo et al. SAC'11
67) Murphy et al. SEHC'11
68) Sun et al. ICWS'11
69) Xie et al. QSIC'11
70) Castro-Cabrera and Medina-Bulo EBT'12
71) Chen et al. ISSDM'12
72) Chen et al. QSIC'12
73) Gagandeep and Singh CCIS'12
74) Liu et al. QSIC'12
75) Pullum and Ozmen BIOMEDCOM'12
76) Ramanathan et al. BIOMEDCOM'12
77) Sun et al. IJWSR'12
78) Xie et al. IST'12
79) Yi et al. ACSIE'12
80) Cao et al. QSIC'13
81) Chan and Tse QSIC'13
82) Dong et al. ICESS'13
83) Hui et al. MPE'13
84) Hui and Huang WCSE'13
85) Hui and Huang WCSE'13 (b)
86) Jiang et al. ICESS'13
87) Kanewala and Bieman ISSRE'13
88) Kanewala and Bieman SECSE'13
89) Lei et al. QSIC'13
90) Rao et al. QSIC'13
91) Yi et al. ISDEA'13
92) Aruna and Prasad ICACCI'14
93) Aruna and Prasad ICT'14
94) Barr et al. TSE'14
95) Carzaniga et al. ICSE'14
96) Goffi et al. FSE'14
97) Goffi ICSEDS'14
98) Kanewala ICSTDS'14
99) Le et al. PLDI'14
100) Liu et al. ICSE'14
101) Liu et al. TSE'14
102) Nuñez and Hierons ATJ'14
103) Segura et al. STVR'14
104) Sun et al. FCS'14
105) Xie et al. QSIC'14
106) Zhang et al. ASE'14
107) Aruna and Prasad ICACCE'15
108) Cañizares et al. ICCS'15
109) Chen AST'15
110) Chen et al. JSS'15
111) Hui et al. STA'15
112) Jameel et al. SNPD'15
113) Jin et al. COMPSAC'15
114) Kanewala et al. STVR'15
115) Lindvall et al. ICSE'15
116) Su et al. AST'15
117) Zhou et al. TSE'15
118) Zhu TSA'15

B.2 Legend

In the following, we detail the meaning of the fields included in the data extraction forms presented in the following pages.

Authors. List of authors' names.

Title. Title of the paper.

Publication. Name of the venue in which the paper was published.

Pub. Type. Type of publication (journal, conference/symposium, workshop or other).

Year. Year of publication (online publication in the case of journal articles).

Pages. Number of pages of the paper.

Country. Affiliation country of the first author of the paper.

Contact. E-mail address of the first author of the paper.

Summary. Short summary of the contributions written by the authors of the review.

Contribution. Type of contribution.

Combination with other techniques. Name of the testing techniques used in combination with metamorphic testing, if any. This does not include the testing techniques used for the generation of source test cases.

Application domain(s). Application domains in which metamorphic testing was applied, e.g., graph theory.

Application scenarios. Specific application scenarios in which metamorphic testing was applied, e.g., shortest path problem.

Number of MRs. Number of metamorphic relations reported on each application scenario.

Program. Name of the program used to evaluate the approach.

Language. Programming language of the subject program.

Size. Number of lines of code of the subject program.

Real. When enabled, it indicates that the program is either commercial or open–source. We did not consider as real those open source programs specifically developed to work as testing benchmarks.

STCs. Number of source test cases used for testing the subject program.

Mutants. Number of artificial faults (i.e., mutants) seeded in the subject program.

Faults. Number of real–world faults uncovered in the program under test.

Source TCs generation technique. Technique(s) used to generate the source test cases.

Evaluation metrics. Name of the metric(s) used to evaluate the effectiveness of metamorphic testing.

Available evaluation material. Enabled if the paper include the evaluation material (source code, mutants, scripts, etc.).

Lesson learned / guidelines. Lessons learned or guidelines reported on the paper.

Challenges. Challenges reported in the paper.

NR. Not Reported.

B.3 Chen et al. TR'98

See legend in page 24 to know the exact meaning of each field.

1998-chen-tr							
Publication data							
Authors:	T. Y. Chen and S. C. Cheung and S. M. Yiu						
Title:	Metamorphic Testing: A New Approach for Generating Next Test Cases						
Publication:	Technical Report HKUST-CS98-01, Department of Computer Science, The Hong Kong University of Science and Technology						
Pub. Type:	☐ Journal ☐ Conference / Symp. ☐ Workshop ☒ Other: Technical Report						
Year:	1998						
DOI/URL:	http://www.cse.ust.hk/~scc/publ/CS98-01-metamorphictesting.pdf						
Pages:	11						
Country:	Australia						
Contact:	scc@cs.ust.hk						
Summary: First paper introducing *metamorphic testing* as a way to create new test cases from successful ones, overcoming the oracle problem. Authors remark the need of combining metamorphic testing with other test case selection strategies. They also mention that metamorphic testing generally requires the use of problem domain knowledge. Four examples are presented, i) Binary search on sorted array, ii) kth occurrence of x in unsorted array, iii) Shortest path in an undirected graph, and iv) Solving a system of linear equations by Gaussian elimination.							
Contribution							
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☐ Case study / application ☐ Assessment ☐ Tool							
Combination with other techniques:							
Application domain(s):	Numerical program, graph theory						
Application scenarios						Number of MRs	
Binary search on sorted array						4	
Kth occurrence of x in unsorted array						3	
Shortest path in an undirected graph						1	
Solving a system of linear equations by Gaussian elimination						1	
Total:						9	
Evaluation							
Program	Language	Size	Real	STCs	Mutants	Faults	
			☐				
			☐				
Total							
Source TCs generation technique:							
Evaluation metrics:							
☐ Available evaluation material							
Lessons learned / guidelines							
- Metamorphic testing generally requires the use of problem domain knowledge							
Challenges							

B.4 Chen et al. COMPSAC'01

See legend in page 24 to know the exact meaning of each field.

2001-chen-compsac						
Publication data						
Authors:	T. Y. Chen and T. H. Tse and Z. Zhou					
Title:	Fault-Based Testing in the Absence of an Oracle					
Publication:	25th Annual International Computer Software and Applications Conference					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2001					
DOI/URL:	http://dx.doi.org/10.1109/CMPSAC.2001.960614					
Pages:	7					
Country:	Australia					
Contact:	tse@csis.hku.hk					
Summary:						
The article proposes to enhance fault-based testing to alleviate the oracle problem using metamorphic testing. Some examples with numerical problems are presented using both real and symbolic inputs. No experiments are reported.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:	Fault-based testing					
Application domain(s):	Numerical programs					
Application scenarios						Number of MRs
Mathematical function						1
Power						1
Compute exponent						1
					Total:	3
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.5 Chen et al. COMPSAC'02

See legend in page 24 to know the exact meaning of each field.

2002-chen-compsac						
Publication data						
Authors:	T. Y. Chen and J. Feng and T. H. Tse					
Title:	Metamorphic Testing of Programs on Partial Differential Equations: a Case Study					
Publication:	26th Annual International Computer Software and Applications Conference					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2002					
DOI/URL:	http://dx.doi.org/10.1109/CMPSAC.2002.1045022					
Pages:	7					
Country:	Australia					
Contact:	tse@csis.hku.hk					
Summary:						
The paper presents a case study on the use of metamorphic testing of programs on partial differential equations. A specific problem is presented and implemented, i.e. distribution of temperatures on a square plate. The authors present 4 test cases using special values and one metamorphic relation. They show how metamorphic testing effectively detects a seeded fault in the program.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:	Special values					
Application domain(s):	Numerical programs (partial differential equations)					
Application scenarios					Number of MRs	
Partial differential equations (distribution of temperatures on a square plate)					1	
				Total:	1	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Partial differential equation	NR	NR	☐	NR	1	0
			☐			
Total						
Source TCs generation technique:	Test suite (special values)					
Evaluation metrics:	NR					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.6 Chen et al. ISSTA'02

See legend in page 24 to know the exact meaning of each field.

2002-chen-issta						
Publication data						
Authors:	T. Y. Chen and T. H. Tse and Z. Zhou					
Title:	Semi-Proving: an Integrated Method Based on Global Symbolic Evaluation and Metamorphic Testing					
Publication:	Proceedings of the 2002 ACM SIGSOFT international symposium on Software testing and analysis					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2002					
DOI/URL:	http://dx.doi.org/10.1145/566171.566202					
Pages:	5					
Country:	Australia					
Contact:	tse@csis.hku.hk					
Summary: The article proposes a semi-proving method combining global symbolic execution and metamorphic testing. The method combines structural information (white-box) when performing global symbolic execution and functional information (black box) when identifying the expected necessary conditions (i.e. metamorphic relations) for correctness. Two examples are presented with numerical programs.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:	Global symbolic execution					
Application domain(s):	Numerical programs					
Application scenarios					Number of MRs	
Numerical median (1 mutant)					1	
Area under a curve (1 mutant)					1	
				Total:	2	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.7 Chen et al. IST'03

See legend in page 24 to know the exact meaning of each field.

2003-chen-ist						
Publication data						
Authors:	T. Y. Chen and T. H. Tse and Z. Zhou					
Title:	Fault-based testing without the need of oracles					
Publication:	Information and Software Technology					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2003					
DOI/URL:	http://dx.doi.org/10.1016/S0950-5849(02)00129-5					
Pages:	9					
Country:	Australia					
Contact:	tychen@it.swin.edu.au					
Summary:						
The article proposes to enhance fault-based testing to alleviate the oracle problem using metamorphic testing. Some examples with numerical problems are presented using both real and symbolic inputs. The authors conclude that different metamorphic relations may have different fault-detection capabilities for different types of faults. This work is an extended version of a conference paper (Chen et al. COMPSAC 2011).						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:	Fault-based testing					
Application domain(s):	Numerical programs					
Application scenarios						Number of MRs
Mathematical function						1
Power						1
Sin						2
Area under a curve#						1
					Total:	5
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
- Different metamorphic relations may have different fault-detection capabilities for different types of faults.						
Challenges						

B.8 Gotlieb and Botella COMPSAC'03

See legend in page 24 to know the exact meaning of each field.

2003-gotlieb-compsac						
Publication data						
Authors:	A. Gotlieb and B. Botella					
Title:	Automated Metamorphic Testing					
Publication:	27th Annual International Computer Software and Applications Conference					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2003					
DOI/URL:	http://dx.doi.org/10.1109/CMPSAC.2003.1245319					
Pages:	7					
Country:	France					
Contact:	Arnaud.Gotlieb@irisa.fr					
Summary:						
The paper presents an Automated Metamorphic Testing (AMT) framework written in Java and Prolog The framework uses constraint programming to find test data that violate certain Metamorphic Relations (MRs). The tool is evaluated using mutation testing on three academic programs written in a subset of C. The types of MRs supported by the tool are limited to numeric expressions over integers.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☒ Tool						
Combination with other techniques:						
Application domain(s):	Numerical programs					

Application scenarios	Number of MRs
Binary search into a sorted array	1
Median	1
Is scalene triangle	2
Total:	4

Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
bsearch	C	17	☐	NR	3	0
GetMid	C	17	☐	NR	2	0
trityp	C	28	☐	NR	33	0
Total		62			39	
Source TCs generation technique:	Constraint programming					
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.9 Chen et al. IBCSE'04

See legend in page 24 to know the exact meaning of each field.

2004-chen-ibcse						
Publication data						
Authors:	T. Y. Chen and D. H. Huang and T. H. Tse and Z. Zhou					
Title:	Case Studies on the Selection of Useful Relations in Metamorphic Testing					
Publication:	Proceedings of the 4th Ibero-American Symposium on Software Engineering and Knowledge Engineering					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2004					
DOI/URL:	http://grise.upm.es/rearviewmirror/conferencias/jiisic04/Papers/25.pdf#sthash.FzIbXIGQ.dpu					
Pages:	15					
Country:	Australia					
Contact:	zhzhou@it.swin.edu.au					
Summary:						
The paper presents two case studies on the selection of useful metamorphic relations. In particular, they compare the effectiveness of MRs identified from a black-box perspective to those obtained using a white-box approach. Several experiments are presented measuring the fault-detection capability of different MRs on two mutated graph-theory programs. Several lessons learned are presented as the main conclusion of the study.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☒ Other: Guidelines ☒ Case study / application ☒ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Graph theory					
Application scenarios						Number of MRs
Shortest path						4
Critical path program						3
					Total:	7
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
ShortestPath			☐	1000	19	
CriticalPath			☐	1000	18	
Total				2000	37	
Source TCs generation technique:	Random					
Evaluation metrics:	Fault detection rate					
☐ Available evaluation material						
Lessons learned / guidelines						
- Theoretical knowledge of the problem domain is not adequate for distinguishing good MRs. - Good MRs should be those that can make the multiple executions of the SUT as different as possible. - Good MRs should be selected with regard to the algorithm that the program follows because algorithms are easier to understand than the source code. - Different MRs have different failure-detecting capabilities with regard to different types of program defect.						
Challenges						
- Prioritize MRs according to their fault detection capability.						

B.10 Chen et al. SNPD'04

See legend in page 24 to know the exact meaning of each field.

2004-chen-snpd						
Publication data						
Authors:	T. Y. Chen and F. Kuo and Y. Liu and A. Tang					
Title:	Metamorphic Testing and Testing with Special Values					
Publication:	Int. Conf. on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2004)					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2004					
DOI/URL:	http://hdl.handle.net/1959.3/3613					
Pages:	7					
Country:	Australia					
Contact:	*dkuo*@it.swin.edu.au					
Summary:						
This paper proposes the use of special values as source test cases for the application of MT. Special test values are test input values for which the expected output is well known. The approach is evaluated with two subject programs using both special values and random values as source test cases. The results show that MT complements and improve the fault-detection effectiveness of special value testing. It also reveal that random testing is an effective approach to augment the number of source test cases and thus to cover more of the test domain.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Numerical programs					
Application scenarios					Number of MRs	
Sin					10	
Binary search					2	
				Total:	12	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Sin	C	41	☐	10	1	0
Binary search	C	43	☐	13	1	0
Total		84		23	2	0
Source TCs generation technique:	Special values and random					
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
- It is useful to perform MT with all the available MRs. - MRs should be "strong". A strong MR should exercise the core functionality of the program. It also should have high sensitivity to fault meaning that the relationship does not hold true for most input data. - Some MRs are more sensitive to faults than others.						
Challenges						

B.11 Chen et al. STEP'04

See legend in page 24 to know the exact meaning of each field.

2004-chen-step							
Publication data							
Authors:	T. Y. Chen and F. Kuo and T. H. Tse and Z. Zhou						
Title:	Metamorphic Testing and Beyond						
Publication:	Eleventh Annual International Workshop on Software Technology and Engineering Practice						
Pub. Type:	☐ Journal ☐ Conference / Symp. ☒ Workshop ☐ Other:						
Year:	2004						
DOI/URL:	http://doi.ieeecomputersociety.org/10.1109/STEP.2003.18						
Pages:	7						
Country:	Australia						
Contact:	tse@csis.hku.hk						
Summary: The paper presents the basic concepts of metamorphic testing and its application illustrating them with examples. Also, some lessons learned and guidelines for the design of effective metamorphic relations are presented.							
Contribution							
☐ New technique / method ☒ Survey / overview ☐ Empirical study ☒ Other: Overview of MT/Guidelines ☐ Case study / application ☐ Assessment ☐ Tool							
Combination with other techniques:							
Application domain(s):	Numerical programs						
Application scenarios							Number of MRs
Sin							10
Partial equation problem							1
Power							1
Med							1
Shortest Path							3
						Total:	16
Evaluation							
Program	Language	Size	Real	STCs	Mutants	Faults	
			☐				
			☐				
Total							
Source TCs generation technique:							
Evaluation metrics:							
☐ Available evaluation material							
Lessons learned / guidelines							
- The failure-causing abilities of different MRs vary greatly. - It is recommended to employ more than one MR due to the previous finding. - Theoretically stronger MRs may not necessarily be more effective in detecting faults than weaker ones. - When selecting MR to test a given program, the algorithm and structure of the algorithm should be taken into account. - MRs that can make the second execution most different from the first one are likely to achieve the best failure-revealing effect.							
Challenges							

- Find out desirable characteristics of MRs that are good at revealing failures.

B.12 Tse et al. COMPSAC'04

See legend in page 24 to know the exact meaning of each field.

2004-tse-compsac						
Publication data						
Authors:	T. H. Tse and S. S. Yau and W. K. Chan and H. Lu and T. Y. Chen					
Title:	Testing Context-Sensitive Middleware-Based Software Applications					
Publication:	28th Annual International Computer Software and Applications Conference					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2004					
DOI/URL:	http://dx.doi.org/10.1109/CMPSAC.2004.1342879					
Pages:	9					
Country:	Hong Kong					
Contact:	thtse@hku.hk					
Summary:						
The paper proposes the application of metamorphic testing for the detection of faults in context-sensitive middleware-based software applications. The authors introduce the topic of context-sensitive applications and present a specific application scenario, i.e. a smart streetlight system. Then, they show how a certain metamorphic relation could help to detect two seeded faults in the sample program.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Embedded systems (context-sensitive middleware-based applications)					
Application scenarios						Number of MRs
Smart streetlight system						1
					Total:	1
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.13 Zhou et al. ISFST'04

See legend in page 24 to know the exact meaning of each field.

2004-zhou-isfst						
Publication data						
Authors:	Z. Zhou and D. H. Huang and T. H. Tse and Z. Yang and H. Huang, T. Y. Chen					
Title:	Metamorphic Testing and Its Applications					
Publication:	Proceedings of the 8th International Symposium on Future Software Technology					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2004					
DOI/URL:	http://www.sea.jp/Events/isfst/ISFST2004/CDROM04/Presented04/2P1-T2/ISFST2004_O346.pdf					
Pages:	6					
Country:	Australia					
Contact:	zhzhou@it.swin.edu.au					
Summary: The paper presents an introduction to metamorphic testing and suggests possible applications in different domains. No experimental evaluation is presented.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Numerical programs, graph theory, computer graphics, compilers, interactive software					

Application scenarios	Number of MRs
Sin	1
Partial differential equations	1
Shortest path problem	2
Pixel display	-
Parallelizing compiler	-
Telephone transaction software	-
Total:	4

Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						

Lessons learned / guidelines

- Good knowledge of the problem domain is necessary for an effective application of MT.

Challenges

B.14 Chan et al. QSIC'05

See legend in page 24 to know the exact meaning of each field.

2005-chan-qsic						
Publication data						
Authors:	W. K. Chan and S. C. Cheung and K. R. P. H. Leung					
Title:	Towards a Metamorphic Testing Methodology for Service-Oriented Software Applications					
Publication:	First International Workshop on Services Engineering					
Pub. Type:	☐ Journal ☐ Conference / Symp. ☒ Workshop ☐ Other:					
Year:	2005					
DOI/URL:	http://dx.doi.org/10.1109/QSIC.2005.67					
Pages:	7					
Country:	Hong Kong					
Contact:	wkchan@cs.ust.hk					
Summary: The paper presents a MT-oriented testing methodology for service-oriented applications. In particular, the authors propose to use so-called metamorphic services to encapsulate services and MRs. The major steps of the methodology are presented for both unit and integration testing. A theoretical illustration example is presented.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Service-oriented software applications					
Application scenarios						Number of MRs
Foreign exchange dealing service						3
					Total:	3
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						
- How to find suitable metamorphic relations for a service?						

B.15 Chan et al. QSIC'05 (b)

See legend in page 24 to know the exact meaning of each field.

2005-chan-qsic-b						
Publication data						
Authors:	W. K. Chan and T. Y. Chen and H. Lu and T. H. Tse and S. S. Yau					
Title:	A Metamorphic Approach to Integration Testing of Context-Sensitive Middleware-Based Applications					
Publication:	Fifth International Conference on Quality Software					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2005					
DOI/URL:	http://dx.doi.org/10.1109/QSIC.2005.3					
Pages:	9					
Country:	Hong Kong					
Contact:	thtse@cs.hku.hk					
Summary: The paper proposes the application of metamorphic testing for the detection of faults in context-sensitive middleware-based software applications. The authors introduce the topic of context-sensitive applications and present a specific application scenario, i.e. a smart delivery system. The paper extends the work of Tse et al. (COMPSAC 2004) to scenarios subjected to evolution. The notion of checkpoint is introduced to facilitate checking the results of MRs.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Embedded systems (context-sensitive middleware-based applications)					
Application scenarios						Number of MRs
Smart delivery system						2
					Total:	2
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.16 Chen et al. WEUSE'05

See legend in page 24 to know the exact meaning of each field.

2005-chen-weuse						
Publication data						
Authors:	T. Y. Chen and F. Kuo and Z. Zhou					
Title:	An Effective Testing Method for End-User Programmers					
Publication:	First workshop on End-user software engineering					
Pub. Type:	☐ Journal ☐ Conference / Symp. ☒ Workshop ☐ Other:					
Year:	2005					
DOI/URL:	http://dx.doi.org/10.1145/1083231.1083236					
Pages:	5					
Country:	Australia					
Contact:	zhzhou@it.swin.edu.au					
Summary: This paper proposes MT as a suitable testing method for end-user programmers. Some sample applications are presented in three different domains: i) Simulation and scientific computation, ii) spreadsheet and DB applications, and iii) web applications. Some lessons learned for the definition of good MRs are presented.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☒ Other: Guidelines ☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Numerical programs, spreadsheet, DB applications, Web application					
Application scenarios						Number of MRs
Thermodynamic problem (partial differential equation)						1
Spreadsheet application						1
Web user interface						
Web user actions (search engine)						
					Total:	2
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
- Good MRs are those that can make multiple executions as different as possible. - Identification of good MRs requires the tester to have both black-box knowledge of the problem domain and white-box knowledge of the program structure.						
Challenges						

B.17 Sim et al. EEEC'05

See legend in page 24 to know the exact meaning of each field.

2005-sim-eeec						
Publication data						
Authors:	K. Y. Sim and W. K. S. Pao and C. Lin					
Title:	Metamorphic testing using geometric interrogation technique and its application					
Publication:	Electrical Engineering/Electronics, Computer, Telecommunications, and Information Technology International Conference.					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2005					
DOI/URL:	http://hdl.handle.net/1959.3/2609					
Pages:	4					
Country:	Malaysia					
Contact:	ksim@swinburne.edu.my					
Summary: This paper presents a metamorphic testing approach for casting simulation using medial axis transform. The authors first present the application scenario and then they introduce 4 sample metamorphic relations. No empirical evaluation is presented.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Simulation					
Application scenarios						Number of MRs
Casting simulation						4
				Total:		
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.18 Tse COMPSAC'05

See legend in page 24 to know the exact meaning of each field.

2005-tse-compsac						
Publication data						
Authors:	T. H. Tse					
Title:	Research Directions in Model-Based Metamorphic Testing and Verification					
Publication:	29th Annual International Computer Software and Applications Conference					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2005					
DOI/URL:	http://dx.doi.org/10.1109/COMPSAC.2005.130					
Pages:	1					
Country:	Hong Kong					
Contact:	thtse@cs.hku.hk					
Summary: The paper briefly presents some research direction in the context of metamorphic testing including model-based metamorphic testing and verification. Some previous contributions of the authors are presented as illustrative examples.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☒ Other: Research directions ☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:	Model-based testing					
Application domain(s):						
Application scenarios						Number of MRs
				Total:		
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.19 Wu COMPSAC'05

See legend in page 24 to know the exact meaning of each field.

2005-wu-compsac						
Publication data						
Authors:	P. Wu					
Title:	Iterative Metamorphic Testing					
Publication:	29th Annual International Computer Software and Applications Conference					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2005					
DOI/URL:	http://dx.doi.org/10.1109/COMPSAC.2005.93					
Pages:	6					
Country:	China					
Contact:	wp@ios.ac.cn					
Summary: This paper proposes applying metamorphic relation iteratively as a way to increase the number of generated test cases and their effectiveness at detecting faults. A case study is presented with a C program for sparse matrix multiplication and more than 1300 mutants. Results reveal that iterative mutation testing outperforms classical metamorphic testing and special case testing in terms of their fault detection capability.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☐ Case study / application ☐ Assessment ☒ Tool						
Combination with other techniques:						
Application domain(s):	Numerical programs					
Application scenarios						Number of MRs
Sparse matrix multiplication						9
					Total:	9
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
SpMatMul (from JASPA benchmark)	C	35	☐	NR	1325	0
			☐			
Total						
Source TCs generation technique:	Test suite					
Evaluation metrics:	Mutation Score (MS) and Fault Detection Ratio (FD)					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						
- Find out the relationships between the number of iterations and the number of faults detected.						

B.20 Wu et al. JS'05

See legend in page 24 to know the exact meaning of each field.

2005-wu-js						
Publication data						
Authors:	P. Wu and X. Shi and J. Tang and H. Lin and T. Y. Chen					
Title:	Metamorphic testing and special case testing: a case study					
Publication:	Journal of Software					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2005					
DOI/URL:	http://dx.doi.org/10.1360/jos161210					
Pages:	11					
Country:	China					
Contact:	wp@ios.ac.cn					
Summary:						
This paper evaluates and compares three testing approaches, namely: i) special case testing, ii) metamorphic testing with special values, and iii) metamorphic testing with random test cases. The effectiveness of the testing methods is evaluated using a subject program of sparse matrix multiplication and mutation analysis. Among other results, the study reveals that metamorphic testing with random test cases is more effective than metamorphic testing with special test cases. It also shows that metamorphic testing and special case testing are complementary methods.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☒ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Numerical program					
Application scenarios						Number of MRs
Sparse matrix multiplication						9
				Total:		9
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
SpMatMult (from JASPA benchmark)	C	35	☐	NR	5	0
			☐			
Total					5	0
Source TCs generation technique:	Special values and random					
Evaluation metrics:	Mutation Score (MS) and Fault Detection Ratio (FDR)					
☐ Available evaluation material						
Lessons learned / guidelines						
- Metamorphic relation selection is crucial to metamorphic testing. - MT with special test cases well supplements the fault detection capabilities of special test cases. - MT with random source test cases outperform that with special test cases.						
Challenges						

B.21 Beydeda COMPSAC'06

See legend in page 24 to know the exact meaning of each field.

2006-beydeda-compsac						
Publication data						
Authors:	S. Beydeda					
Title:	Self-Metamorphic-Testing Components					
Publication:	The Third International Workshop on Software Cybernetics					
Pub. Type:	☐ Journal ☐ Conference / Symp. ☒ Workshop ☐ Other:					
Year:	2006					
DOI/URL:	http://dx.doi.org/10.1109/COMPSAC.2006.161					
Pages:	6					
Country:	Germany					
Contact:	sb@stecc.de					
Summary: This paper proposes integrating self-testing capabilities in COST components using MRs. A very preliminary case study is presented. No MRs are presented.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Components					
Application scenarios						Number of MRs
				Total:		
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.22 Chan et al. IJSEKE'06

See legend in page 24 to know the exact meaning of each field.

2006-chan-ijseke						
Publication data						
Authors:	W.K. Chan and T.Y. Chen and H. Lu and T.H. Tse and S.S. Yau					
Title:	Integration Testing of Context-Sensitive Middleware-Based Applications: a Metamorphic Approach					
Publication:	International Journal of Software Engineering and Knowledge Engineering					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2006					
DOI/URL:	https://dx.doi.org/10.1142/S0218194006002951					
Pages:	25					
Country:	Hong Kong					
Contact:	thtse@cs.hku.hk					
Summary:						
The paper proposes the application of metamorphic testing for the detection of faults in context-sensitive middleware-based software applications. The authors introduce the topic of context-sensitive applications and present a specific application scenario, i.e. a smart delivery system. The approach is illustrated with an experiment on the detection of faults in an RFID-based location estimation program running on a context-aware prototype. This work is an extension of a conference paper (Chan et al. QSIC 2005)						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Embedded systems (context-sensitive middleware-based applications)					
Application scenarios					Number of MRs	
Smart delivery system					2	
				Total:	2	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Cabot system v2.0	Java	NR	☐	60	21	
			☐			
Total				60	21	
Source TCs generation technique:	Random testing					
Evaluation metrics:	Mutation score					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.23 Hu et al. SOQUA'06

See legend in page 24 to know the exact meaning of each field.

2006-hu-soqua						
Publication data						
Authors:	P. Hu and Z. Zhang W. K. Chan and T. H. Tse					
Title:	An Empirical Comparison between Direct and Indirect Test Result Checking Approaches					
Publication:	Third International Workshop on Software Quality Assurance					
Pub. Type:	☐ Journal ☐ Conference / Symp. ☒ Workshop ☐ Other:					
Year:	2006					
DOI/URL:	http://dx.doi.org/10.1145/1188895.1188901					
Pages:	8					
Country:	Hong Kong					
Contact:	thtse@cs.hku.hk					
Summary: This paper reports on a controlled experiment to investigate the cost effectiveness of using MT by 38 testers on three open-source programs. The results are compared with those of assertion checking. The results suggest that MT is more effective than assertion checking in detecting faults but it is more time consuming. Several lessons learned are presented.						
Contribution						
☐ New technique / method ☐ Survey / overview ☒ Empirical study ☐ Other: ☐ Case study / application ☒ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Text patterns, Boolean expression evaluation, office application					
Application scenarios					Number of MRs	
Text pattern search					18	
Boolean expression evaluation					39	
Table sorting					25	
				Total:	82	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Boyer - Pattern search	Java	241	☒	NR	132	0
Jboolexpr - Boolean expression	Java	231	☒	NR	127	0
Eurobadget - TxnTableSorter	Java	281	☒	NR	317	0
Total		753			576	0
Source TCs generation technique:	Test suite					
Evaluation metrics:	Mutation score					
☐ Available evaluation material						
Lessons learned / guidelines						
- The more MRs are used, the higher the mutation detection ratio. - The effectiveness of using a MR increases as we increase the number of test cases.						
Challenges						
- There is a need to propose more systematic methods for creating metamorphic relations. - It is necessary to know which MRs should be given a higher priority.						

B.24 Mayer and Guderlei COMPSAC'06

See legend in page 24 to know the exact meaning of each field.

2006-mayer-compsac						
Publication data						
Authors:	J. Mayer and R. Guderlei					
Title:	An Empirical Study on the Selection of Good Metamorphic Relations					
Publication:	30th Annual International Computer Software and Applications Conference					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2006					
DOI/URL:	http://dx.doi.org/10.1109/COMPSAC.2006.24					
Pages:	10					
Country:	Germany					
Contact:	johannes.mayer@uni-ulm.de					
Summary: This paper presents an empirical assessment of the quality of MRs. Six Java programs for determinant computation are mutated and used as a case study. The authors presents 16 MRs and apply then to randomly generated test cases checking the number of killed mutants. As a result, a number of rules for judging the suitability of MRs are reported.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☒ Other: Guidelines ☐ Case study / application ☒ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Numerical programs					
Application scenarios					Number of MRs	
Determinant computation					16	
				Total:	16	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Commons.Math Apache v1.0	Java	NR	☒	100K/ mutants	149	0
JScience v2.0.1	Java	NR	☒	100K/ mutants	1	0
JAMA v1.0.2	Java	NR	☒	100K/ mutants	76	0
Impl. of Michael Flanagan 01/05/2005	Java	NR	☒	100K/ mutants	183	0
Impl. of Jon Squire 20/10/2005	Java	NR	☒	100K/ mutants	60	0
GeoStoch	Java	NR	☒	100K/ mutants	59	0
Total					528	0
Source TCs generation technique:	Random					
Evaluation metrics:	Number of test cases to kill a mutant					
☐ Available evaluation material						
Lessons learned / guidelines						

- MRs that have the form of equalities are especially weak. - Equalities of linear combinations are stronger than simple equalities. - Good MRs contain much of the semantics of the SUT. - MRs should not be close to the implementation/algorithm under test. - Combining MRs may yield better results than applying the MRs independently (at the expense of higher costs)
Challenges

B.25 Mayer and Guderlei QSIC'06

See legend in page 24 to know the exact meaning of each field.

2006-mayer-qsic							
Publication data							
Authors:	J. Mayer and R. Guderlei						
Title:	On Random Testing of Image Processing Applications						
Publication:	Sixth International Conference on Quality Software						
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:						
Year:	2006						
DOI/URL:	http://dx.doi.org/10.1109/QSIC.2006.45						
Pages:	8						
Country:	Germany						
Contact:	johannes.mayer@uni-ulm.de						
Summary: This paper proposes an approach for random testing of image processing application using metamorphic testing. Two models for random generation are evaluated using mutation testing and several MRs. Also, some properties and special values are proposed. The approach is evaluated using a Java implementation of the Euclidean distance transform integrated as a part of the GeoStoch library.							
Contribution							
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☒ Case study / application ☐ Assessment ☐ Tool							
Combination with other techniques:							
Application domain(s):	Computer graphics						
Application scenarios						Number of MRs	
Euclidean distance transform (image processing)						7	
					Total:	7	
Evaluation							
Program	Language	Size	Real	STCs	Mutants	Faults	
Euclidean distance (GeoStoch library)	Java	NR	☒	1000	1334		
Total				1000	1334		
Source TCs generation technique:	Random, special values						
Evaluation metrics:	Number of mutants killed by each MR						
☐ Available evaluation material							
Lessons learned / guidelines							
- Special value testing should be accompanied by another testing strategy. - Combined application of MRs yield better results than MRs in isolation.							
Challenges							

B.26 Chan et al. COMPSAC'07

See legend in page 24 to know the exact meaning of each field.

2007-chan-compsac						
Publication data						
Authors:	W. K. Chan and J. C. F. Ho and T. H. Tse					
Title:	Piping Classification to Metamorphic Testing: An Empirical Study towards Better Effectiveness for the Identification of Failures in Mesh Simplification Programs					
Publication:	31st Annual International Computer Software and Applications Conference					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2007					
DOI/URL:	http://dx.doi.org/10.1109/COMPSAC.2007.167					
Pages:	8					
Country:	Hong Kong					
Contact:	thtse@cs.hku.hk					
Summary: This paper presents a testing approach for mesh simplification programs using pattern classification and metamorphic testing. First, test cases are classified as passed or failed by a pattern classification component. Then, metamorphic testing is used to detect missed failures in those test cases classified as passed. A case study with four java programs and several hundreds of mutants are presented. Three MRs are used. The experimental results reveal a 10% improvement in effectiveness.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Computer graphics					
Application scenarios						Number of MRs
Euclidean distance transform (image processing)						3
					Total:	3
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Shortest	Java	NR	☐	10648	350	0
Melax	Java	NR	☐	10648	401	0
Quadric	Java	NR	☐	10648	1122	0
QuadricTri	Java	NR	☐	10648	1187	0
Total				42592	3060	
Source TCs generation technique:	Test suite					
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.27 Chan et al. IJWSR'07

See legend in page 24 to know the exact meaning of each field.

2007-chan-ijwsr							
Publication data							
Authors:	W. K. Chan and S. C. Cheung and K. R. P. H. Leung						
Title:	A Metamorphic Testing Approach for Online Testing of Service-Oriented Software Applications						
Publication:	International Journal of Web Services Research						
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:						
Year:	2007						
DOI/URL:	http://www.cs.cityu.edu.hk/~wkchan/papers/sacmta09-chan+cheung+leung.pdf						
Pages:	21						
Country:	Hong Kong						
Contact:	wkchan@cs.cityu.edu.hk						
Summary: The paper presents a MT-oriented testing methodology for service-oriented applications. The authors propose to use so-called metamorphic services to encapsulate services and MRs. An experiment with a service-oriented calculator is presented. The results reveal higher effectiveness with less effort, compared to a control experiment not using MT. The work is an extension of a conference paper (Chan et al. 2005 QSIC).							
Contribution							
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☒ Case study / application ☐ Assessment ☐ Tool							
Combination with other techniques:							
Application domain(s):	Service-oriented software applications						
Application scenarios						Number of MRs	
Foreign exchange dealing service						3	
Service oriented calculator						3+	
					Total:	6+	
Evaluation							
Program	Language	Size	Real	STCs	Mutants	Faults	
Service oriented calculator	C++	2480	☐	25006	6		
Total		2480			6		
Source TCs generation technique:	Test suite (black-box combinatorial approach)						
Evaluation metrics:							
☐ Available evaluation material							
Lessons learned / guidelines							
Challenges							

B.28 Chan et al. RST'07

See legend in page 24 to know the exact meaning of each field.

2007-chan-rst						
Publication data						
Authors:	W. K. Chan and T. Y. Chen and S. C. Cheung and T. H. Tse and Z. Zhang					
Title:	Towards the Testing of Power-Aware Software Applications for Wireless Sensor Networks					
Publication:	12th Ada-Europe International Conference on Reliable Software Technologies					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2007					
DOI/URL:	http://dx.doi.org/10.1007/978-3-540-73230-3_7					
Pages:	16					
Country:	Hong Kong					
Contact:	wkchan@cs.cityu.edu.hk					
Summary:						
This paper proposes the application of MT to Wireless Sensor Networks (WSN) software systems. As a novelty, authors propose testing non-functional properties related to power consumption. A temperature monitoring application scenario is used to illustrate the approach.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Embedded systems (wireless sensor network applications)					
Application scenarios						Number of MRs
Temperature monitoring						2
				Total:		2
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.29 Dong et al. QSIC'07

See legend in page 24 to know the exact meaning of each field.

2007-dong-qsic						
Publication data						
Authors:	G. Dong and C. Nie and B. Xu and L. Wang					
Title:	An Effective Iterative Metamorphic Testing Algorithm Based on Program Path Analysis					
Publication:	Seventh International Conference on Quality Software					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2007					
DOI/URL:	http://dx.doi.org/10.1109/QSIC.2007.4385510					
Pages:	6					
Country:	China					
Contact:	dgw@seu.edu.cn					
Summary:						
This paper presents an algorithm for iterative MT named APCEMSI. The idea is to apply MRs iteratively as proposed by Wu (Wu, COMPSAC 2005) until a path coverage criterion is fulfilled, namely, APCEM (All-Path Coverage for Every MR). A small experiment is presented evaluating the effectiveness of the algorithm with 4 mutants and 7 MRs.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:	Symbolic execution, structural testing					
Application domain(s):	Numerical programs					
Application scenarios						Number of MRs
TriSquare. Check whether 3 positive real numbers could construct a triangle						7
					Total:	7
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
TriSquare	Java	30	☐	100	4	
			☐			
Total				100	4	
Source TCs generation technique:						
Evaluation metrics:	Mutation score, Faults on problem Path Detection ratio (FPD), MR Detection Performance (MDP), MR Detection ratio for each Mutant (MDM)					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.30 Guderlei and Mayer IJSEKE'07

See legend in page 24 to know the exact meaning of each field.

2007-guderlei-ijseke	
Publication data	
Authors:	R. Guderlei and J. Mayer
Title:	Towards automatic testing of imaging software by means of random and metamorphic testing.
Publication:	International Journal of Software Engineering and Knowledge Engineering
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:
Year:	2007
DOI/URL:	https://dx.doi.org/10.1142/S0218194007003471
Pages:	25
Country:	Germany
Contact:	ralph.guderlei@uni-ulm.de

Summary:

This article proposes an approach for random testing of image processing application using metamorphic testing. Two models for random generation are evaluated using mutation testing and several MRs. In particular, two types of MRs are presented: 4 general MRs applicable to most image operators and 5 MRs specifically designed for the Euclidean distance transform operator. Also, some properties and special values are proposed. The approach is evaluated using three Java implementations of different image operators. This work is an extension of a conference paper (Mayer and Guderlei QSIC 2006)

Contribution

☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:

☒ Case study / application ☐ Assessment ☐ Tool

Combination with other techniques:	
Application domain(s):	Computer graphics

Application scenarios	Number of MRs
General image processing	4
Euclidean distance transform (image processing)	5
Total:	9

Evaluation

Program	Language	Size	Real	STCs	Mutants	Faults
GeoStoch library	Java	301	☒	1000	1241	
ConnectedC8	Java	215	☐	1000	495	
Lipschitz	Java	320	☒	100	940	
Total		836		2100	2676	

Source TCs generation technique:	Random testing and special values
Evaluation metrics:	Number of mutants killed by each MR

☐ Available evaluation material

Lessons learned / guidelines

- Special value testing should be accompanied by another testing strategy.

Challenges

B.31 Guderlei and Mayer QSIC'07

See legend in page 24 to know the exact meaning of each field.

2007-guderlei-qsic						
Publication data						
Authors:	R. Guderlei and J. Mayer					
Title:	Statistical Metamorphic Testing – Testing Programs with Random Output by Means of Statistical Hypothesis Tests and Metamorphic Testing					
Publication:	First International Workshop on Software Test Evaluation					
Pub. Type:	☐ Journal ☐ Conference / Symp. ☒ Workshop ☐ Other:					
Year:	2007					
DOI/URL:	http://dx.doi.org/10.1109/QSIC.2007.4385527					
Pages:	6					
Country:	Germany					
Contact:	ralph.guderlei@uni-ulm.de					
Summary:						
This paper presents a new testing method for non-deterministic programs called Statistical Metamorphic Testing (SMT). In SMT, two or more independent output sequences are generated and then compared according to MRs using statistical hypothesis tests. A small case study is presented. Although the effectiveness of the approach is not demonstrated, the authors claim that their approach is the only approach to test randomized software where not theoretical values about the output distributions are known.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:	Statistical hypothesis testing					
Application domain(s):	Simulation					
Application scenarios						Number of MRs
Simulation algorithm for random mosaics						2
Inverse cumulative distribution function of the normal distribution Φ^{-1}						1
					Total:	3
Evaluation						
Program	Language	Size	Real	STCs	Mutant	Faults
Inverse cumulative distribution function of the normal distribution Φ^{-1}	NR	90	☐	5000/ mutant	306	0
			☐			
Total		90			306	
Source TCs generation technique:	Random					
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.32 Zhou et al. TR'07

See legend in page 24 to know the exact meaning of each field.

2007-zhou-tr						
Publication data						
Authors:	Z. Zhou, T. H. Tse, F.-C. Kuo and T. Y. Chen					
Title:	Automated Functional Testing of Web Search Engines in the Absence of an Oracle					
Publication:	Technical report – Department of Computer Science, The University of Hong Kong					
Pub. Type:	☐ Journal ☐ Conference / Symp. ☐ Workshop ☒ Other:TR					
Year:	2007					
DOI/URL:	http://www.cs.hku.hk/research/techreps/document/TR-2007-06.pdf					
Pages:	12					
Country:	Australia					
Contact:	zhiquan@uow.edu.au					
Summary:						
This technical report presents a case study and an associated tool for metamorphic testing of web search engines. Several metamorphic relations are presented and illustrated with examples in three real search engines: Google, Yahoo and LiveSearch. An automated testing tool is also presented and evaluated on these search engines in which several failures were revealed. An extension of this work was published in the STVR journal in 2010.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☒ Tool						
Combination with other techniques:						
Application domain(s):	Search engines					
Application scenarios						Number of MRs
Web search						9
				Total:		9
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Google			☒			3
Yahoo			☒			1
LiveSearch			☒			2
Total						6
Source TCs generation technique:	Random (using a dictionary)					
Evaluation metrics:	Failure rate					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.33 Dong et al. JSU'08

See legend in page 24 to know the exact meaning of each field.

2008-dong-jsu						
Publication data						
Authors:	G. Dong and B. Xu and L. Chen and C. Nie and L. Wang					
Title:	Case studies on testing with compositional metamorphic relations					
Publication:	Journal of Southeast University					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2008					
DOI/URL:	http://caod.oriprobe.com/articles/15290800/Case_studies_on_testing_with_compositional_metamorphic_relations.htm					
Pages:	6					
Country:	China					
Contact:	bwxu@seu.edu.cn					
Summary:						
The authors propose to create new metamorphic relations by composing existing relations with the aim of improving their fault detection capability and reduce the number of executed test cases. The approach is evaluated with to small case studies from which a few lessons learned are reported. This method was later explored in more detail by Liu et al (2012-liu-qsic).						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Numerical programs					
Application scenarios					Number of MRs	
Sparse matrix multiplication					9	
Triangle square calculation					7	
				Total:	16	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Sparse matrix multiplication	C	26	☐	8	5	
Triangle square calculation	C	13	☐	5	4	
Total		39		13	9	
Source TCs generation technique:	Special values / existing suite					
Evaluation metrics:	Mutation score, fault detection ratio					
☐ Available evaluation material						
Lessons learned / guidelines						
- The order of the composition matters. - Not all metamorphic relations can be composed.						
Challenges						

B.34 Murphy FSEDS'08

See legend in page 24 to know the exact meaning of each field.

2008-murphy-fseds						
Publication data						
Authors:	C. Murphy					
Title:	Using Runtime Testing to Detect Defects in Applications without Test Oracles					
Publication:	Foundations of Software Engineering Doctoral Symposium					
Pub. Type:	☐ Journal ☐ Conference / Symp. ☐ Workshop ☒ Other: Doctoral symposium					
Year:	2008					
DOI/URL:	http://dx.doi.org/10.1145/1496653.1496659					
Pages:	4					
Country:	United States					
Contact:	cmurphy@cs.columbia.edu					
Summary: The paper was presented in a doctoral symposium and anticipates the thesis contribution of the authors. In particular, the author proposes using runtime MT for the detection of faults in highly configurable systems.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):						
Application scenarios						Number of MRs
					Total:	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.35 Murphy et al. TR'08

See legend in page 24 to know the exact meaning of each field.

2008-murphy-tr						
Publication data						
Authors:	C. Murphy and G. Kaiser and L. Hu					
Title:	Properties of Machine Learning Applications for Use in Metamorphic Testing					
Publication:	Department of Computer Science, Columbia University, New York NY					
Pub. Type:	☐ Journal ☐ Conference / Symp. ☐ Workshop ☒ Other: Technical Report					
Year:	2008					
DOI/URL:	http://mice.cs.columbia.edu/getTechreport.php?techreportID=509					
Pages:	7					
Country:	United States					
Contact:	cmurphy@cs.columbia.edu					
Summary:						
This paper proposes using MT to alleviate the oracle problem in machine learning applications. To that purpose, the authors define 6 MRs for supervised and unsupervised machine learning algorithms and assess their applicability in three specific tools. They argue that the proposed MRs are generic enough to be applied to other machine learning applications: additive, multiplicative, permutative, invertive, inclusive, and exclusive. They conclude that MT is a suitable and generic approach to address the oracle problem in the machine learning domain.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Machine learning					
Application scenarios						Number of MRs
(Un)supervised ML algorithm						6
				Total:		6
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
MartiRank	NR	NR	☐	NR	0	1
SVM-Light	NR	NR	☐	NR	0	1
PAYL	NR	NR	☐	NR	0	2
Total						
Source TCs generation technique:	Random					
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.36 Chan et al. STVR'09

See legend in page 24 to know the exact meaning of each field.

2009-chan-stvr						
Publication data						
Authors:	W. K. Chan and J. C. F. Ho and T. H. Tse					
Title:	Finding failures from passed test cases: improving the pattern classification approach to the testing of mesh simplification programs					
Publication:	Software Testing, Verification and Reliability Journal					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2009					
DOI/URL:	http://dx.doi.org/10.1002/stvr.v20:2					
Pages:	32					
Country:	Hong Kong					
Contact:	wkchan@cs.cityu.edu.hk					
Summary:	This article presents a testing approach for mesh simplification programs using pattern classification and metamorphic testing. Test cases are first classified as passed or failed by a pattern classification component and then MT is used to detect missed failures in those test cases classified as passed. A case study with three java programs and several hundreds of mutants are presented. Three MRs are used. The article is an extension of a conference paper (Chan et al. 2007 COMPSAC).					
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Computer graphics					
Application scenarios						Number of MRs
Mesh simplification						3
				Total:		3
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Melax	Java	NR	☐	10648	401	0
Quadric	Java	NR	☐	10648	1122	0
QuadricTri	Java	NR	☐	10648	1187	0
Total				31944	2710	
Source TCs generation technique:	Test suite					
Evaluation metrics:	Data mining specifics					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.37 Chen et al. BIOINFORMATICS'09

See legend in page 24 to know the exact meaning of each field.

2009-chen-bioinformatics						
Publication data						
Authors:	T. Y. Chen and J. W. K. Ho and H. Liu and X. Xie					
Title:	An innovative approach for testing bioinformatics programs using metamorphic testing					
Publication:	BioMed Central Bioinformatics Journal					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2009					
DOI/URL:	http://dx.doi.org/10.1186/1471-2105-10-24					
Pages:	12					
Country:	Australia					
Contact:	tychen@swin.edu.au					
Summary:						
The article proposed using MT for the detection of faults in bioinformatics programs with the oracle problem. For the evaluation of the approach, the authors propose 19 MRs for two open-source bioinformatics programs and measure their effectiveness at detecting faults using mutation testing. Random and real inputs are used for the source test cases. Finally, they also mention how MT could be applied to test programs from other domains of bioinformatics.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Bioinformatics					
Application scenarios					Number of MRs	
Network simulation (graph computation)					10	
Approximate string matching problem					9	
				Total:	19	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
GNLab	NR	NR	☒	NR	9	0
SeqMap	NR	NR	☒	NR	3	0
Total					12	
Source TCs generation technique:	Random, tool-based (GRN and E.coli GRN)					
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
- MT can be combined with special values. - MT allows the use of real inputs as test cases. - MT is suitable for bioinformatics programmers. - MT is useful for testing diverse types of programs. - Selecting the most effective MRs requires good understanding of the problem domains. - The effectiveness of MT depends on the number and variety of source test cases.						
Challenges						

B.38 Chen et al. FTDS'09

See legend in page 24 to know the exact meaning of each field.

2009-chen-ftds						
Publication data						
Authors:	T. Y. Chen and F. Kuo and H. Liu and S. Wang					
Title:	Conformance Testing of Network Simulators Based on Metamorphic Testing Technique					
Publication:	Joint 11th IFIP WG 6.1 International Conference FMOODS 2009 and 29th IFIP WG 6.1 International Conference FORTE 2009 on Formal Techniques for Distributed Systems					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2009					
DOI/URL:	http://dx.doi.org/10.1007/978-3-642-02138-1_19					
Pages:	6					
Country:	Australia					
Contact:	hliu@swin.edu.au					
Summary: This paper proposes the application of MT for conformance testing of network simulators. A case study is presented testing the OMNeT++ tool for conformance with the Ad-hoc On-demand Distance Vector (AODV) protocol. Eleven MRs are defined and applied to the program with six seeded faults. The results show a significant success rate in detecting faults.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Simulation					
Application scenarios						Number of MRs
Ad-hoc On-demand Distance Vector (AODV)						11
				Total:		11
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
OMNeT++	C++	NR	☒	NR	6	0
			☐			
Total					6	
Source TCs generation technique:	Random					
Evaluation metrics:	Mutation score					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.39 Chen et al. ICECCS'09

See legend in page 24 to know the exact meaning of each field.

2009-chen-iceccs						
Publication data						
Authors:	T. Y. Chen and F. Kuo and R. Merkel and W. K. Tam					
Title:	Testing an Open Source Suite for Open Queuing Network Modelling using Metamorphic Testing Technique					
Publication:	14th IEEE International Conference on Engineering of Complex Computer Systems					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2009					
DOI/URL:	http://dx.doi.org/10.1109/ICECCS.2009.28					
Pages:	7					
Country:	Australia					
Contact:	dkuo@groupwise.swin.edu.au					
Summary:						
This paper proposes using MT for the detection of faults in open Queuing Network Modelling (QNM) systems. A case study is presented with the JMVA module of JMT open source tool for QNM. In particular, 7 MRs were devised and evaluated using mutation testing. The results suggest that MT is an effective approach for the detection of faults in QNM applications (16 out of 20 mutants were detected)						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Queuing network modelling					
Application scenarios						Number of MRs
Open queuing network modelling						7
				Total:		7
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
JMT – JMVA module v0.7.3	Java	NR	☒	100	20	0
			☐			
Total				100	20	
Source TCs generation technique:	Random					
Evaluation metrics:	Percentage of test cases that detected a mutant M using metamorphic relation MR.					
☐ Available evaluation material						
Lessons learned / guidelines						
- The best results are likely to be achieved by broadest range of MRs.						
Challenges						

B.40 Just and Schweiggert ICSTW'09

See legend in page 24 to know the exact meaning of each field.

2009-just-icstw						
Publication data						
Authors:	R. Just and F. Schweiggert					
Title:	Evaluating testing strategies for imaging software by means of Mutation Analysis					
Publication:	Mutation					
Pub. Type:	☐ Journal ☐ Conference / Symp. ☒ Workshop ☐ Other:					
Year:	2009					
DOI/URL:	http://dx.doi.org/10.1109/ICSTW.2009.20					
Pages:	5					
Country:	Germany					
Contact:	rene.just@uni-ulm.de					
Summary:						
The paper proposes using mutation testing for the selection of suitable test inputs and the evaluation of partial oracles. A case study is presented using MT in an open source library for image processing. Among other results, the authors propose using combinations of MRs to increase the number of mutants detected.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☒ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Computer graphics					
Application scenarios						Number of MRs
JPEG Image decoder						4
				Total:		4
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
JJ2000 library	Java	NR	☒	NR	514	0
			☐			
Total					514	
Source TCs generation technique:	Random					
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.41 Murphy et al. ICST'09

See legend in page 24 to know the exact meaning of each field.

2009-murphy-icst						
Publication data						
Authors:	C. Murphy and K. Shen and G. Kaiser					
Title:	Using JML Runtime Assertion Checking to Automate Metamorphic Testing in Applications without Test Oracles					
Publication:	Second International Conference on Software Testing Verification and Validation					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2009					
DOI/URL:	http://dx.doi.org/10.1109/ICST.2009.19					
Pages:	10					
Country:	United States					
Contact:	cmurphy@cs.columbia.edu					
Summary:						
The paper proposes specifying MRs as runtime assertions for ensuring that the specifications holds during program execution. The author presents a tool called Corduroy that acts as a pre-processor that convert the specification of MRs into JML (Java Modelling Language) assertions. A case study with two real world machine learning tools is presented. The author mentions the use of 25 MRs but they are not reported in the paper. Three real faults were detected.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☒ Tool						
Combination with other techniques:						
Application domain(s):	Machine learning					
Application scenarios					Number of MRs	
Naïve Bayes						
Support vector machines						
K-nearest neighbours						
C4.5						
				Total:	25	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Weka 3.5.8	Java	NR	☒	NR	0	2
RapidMiner 4.1	Java	NR	☒	NR	0	1
Total						3
Source TCs generation technique:	Test suite (UC-Irvine Machine Learning Repository)					
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.42 Murphy et al. ISSTA'09

See legend in page 24 to know the exact meaning of each field.

2009-murphy-issta						
Publication data						
Authors:	C. Murphy and K. Shen and G. Kaiser					
Title:	Automatic System Testing of Programs without Test Oracles					
Publication:	The eighteenth International Symposium on Software Testing and Analysis					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2009					
DOI/URL:	http://dx.doi.org/10.1145/1572272.1572295					
Pages:	11					
Country:	United States					
Contact:	cmurphy@cs.columbia.edu					
Summary:						
This paper presents a framework called Amsterdam for the automated application of MT. The tool takes as inputs the program under test and a set MRs, defined in a XML file. Then, Amsterdam automatically runs the program, applies the MRs and checks the results. In certain cases, the results of two executions may not match due to floating point calculation or non-determinism. To address this issue, the authors propose the concept of "heuristic test oracles", by defining a function that determines whether the outputs are "close enough" to be considered equal. An experimental evaluation is reported with three machine learning applications. The paper is an extension of a previous work (Murphy et al. 2008 Tech report)						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☒ Tool						
Combination with other techniques:						
Application domain(s):	Machine learning					
Application scenarios						Number of MRs
SVM – Classification algorithm						4
C4.5 – decision tree algorithm						4
MartiRank – ranking algorithm						4
PAYL – Unsupervised algorithm						2
					Total:	14
Evaluation						
Program	Language	Size	Real	STCs	Mutant	Faults
SVM-Weka 3.5.8	Java	NR	☒	150	85	0
C4.5	C	NR	☐	150	28	0
MartiRank	NR	NR	☐	10000	69	0
PAYL	NR	NR	☐	NR	40	0
Total					222	
Source TCs generation technique:	Test suite ("iris" dataset from UC-Irvine repository)					
Evaluation metrics:	Mutation score					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						
- The transformation of input data can be laborious and error-prone (large tables, binary files...) - Generation of initial test cases. - Check "close enough" expected solutions, e.g. imprecisions with floating point.						

B.43 Xie et al. QSIC'09

See legend in page 24 to know the exact meaning of each field.

2009-xie-qsic						
Publication data						
Authors:	X. Xie and J. Ho and C. Murphy and G. Kaiser and B. Xu and T. Y. Chen					
Title:	Application of Metamorphic Testing to Supervised Classifiers					
Publication:	Ninth International Conference on Quality Software					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2009					
DOI/URL:	http://dx.doi.org/10.1109/QSIC.2009.26					
Pages:	10					
Country:	Australia					
Contact:	cmurphy@cs.columbia.edu					
Summary:						
These authors propose using MT for the detection of faults in supervised classifiers. They argue that MT can be helpful for both validation and verification. Validation is used to find out whether the algorithm is appropriate for the problem. Verification is used to detect fault in the algorithms. Two specific algorithms are studied: K-Nearest Neighbours (KNN) and Naïve Bayes Classifier (NBC). As a first step, 11 MRs are proposed and applied to implementations of the algorithms in the tool Weka. The results reveal that 5 MRs do not hold for KNN (3 for NBC) and are therefore not necessary properties for the algorithms under study. The rest of MRs, however, show to be effective and detect several defects. This work is an extension of a previous technical report (Murphy 2008 TR).						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Machine learning					
Application scenarios						Number of MRs
K-Nearest neighbours						11
Naïve Bayes Classifier						
					Total:	11
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Weka 3.5.7	Java	NR	☒	NR	0	3
			☐			
Total						3
Source TCs generation technique:	Random					
Evaluation metrics:	Percentage of test cases violating each MR					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.44 Zhang et al. JS'09

See legend in page 24 to know the exact meaning of each field.

2009-zhang-js						
Publication data						
Authors:	Z. Y. Zhang and W. K. Chan and T. H. Tse and P. F. Hu					
Title:	Experimental study to compare the use of metamorphic testing and assertion checking					
Publication:	Journal of Software					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2009					
DOI/URL:	http://dx.doi.org/10.3724/SP.J.1001.2009.00578					
Pages:	19					
Country:	Hong Kong					
Contact:	wkchan@cs.cityu.edu.hk					
Summary:						
This article presents a controlled experiment to compare the cost effectiveness of metamorphic testing and assertion checking. The study was conducted with 38 subject participants and three real-world subject programs. The experiment revealed that metamorphic testing is less efficient but more effective than assertion checking. It also shows that average programmers are able to design and implement metamorphic relations after a general introduction to the technique. Several lessons learned and challenges are reported.						
Contribution						
☐ New technique / method ☐ Survey / overview ☒ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Pattern matching, Boolean expressions, table sorting					
Application scenarios					Number of MRs	
Pattern matching					18	
Boolean expressions					39	
Table sorting					25	
				Total:	82	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Boyer	Java	241	☒	NR	151	
BooleanExpression (jboolexpr)	Java	231	☒	NR	145	
TxnTableSorter	Java	281	☒	NR	378	
Total		753			674	
Source TCs generation technique:	NR					
Evaluation metrics:	Mutation score					
☐ Available evaluation material						
Lessons learned / guidelines						
- The more MRs being used, the higher will be the mutation detection ratio. - The utilization of an MR implementation increases as testers increase the number of initial test cases applicable to the MR. - MRs have different fault detection capability.						
Challenges						
- There is a need to develop systematic methods for creating metamorphic relations.						

B.45 Chen SOSE'10

See legend in page 24 to know the exact meaning of each field.

2010-chen-sose						
Publication data						
Authors:	T. Y. Chen					
Title:	Metamorphic Testing: A Simple Approach to Alleviate the Oracle Problem					
Publication:	Fifth International Symposium on Service Oriented System Engineering					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2010					
DOI/URL:	http://dx.doi.org/10.1109/SOSE.2010.31					
Pages:	2					
Country:	Australia					
Contact:	tychen@swin.edu.au					
Summary: The paper is a two-pages summary of a tutorial on metamorphic testing.						
Contribution						
☐ New technique / method ☒ Survey / overview ☐ Empirical study ☐ Other: ☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):						
Application scenarios						Number of MRs
				Total:		
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.46 Chen et al TSE'10

See legend in page 24 to know the exact meaning of each field.

2010-chen-tse						
Publication data						
Authors:	T. Y. Chen and T.H. Tse and Z. Zhou					
Title:	Semi-Proving: An Integrated Method for Program Proving, Testing, and Debugging					
Publication:	Transactions on Software Engineering Journal					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2010					
DOI/URL:	http://dx.doi.org/10.1109/TSE.2010.23					
Pages:	17					
Country:	Australia					
Contact:	tychen@swin.edu.au					
Summary:						
The article proposes combining MT and symbolic execution in an integrated approach for proving, testing and debugging. The method first proves that the program satisfies certain MRs for the entire input domain or a subset of it, identifying all the inputs that violate the MRs. For certain programs, the method can be also turned into a conventional symbolic-testing approach, testing a subset of selected paths. The approach also supports automated debugging through the identification of constraint expressions that reveal failures. A case study with one of the C programs of the Siemens Suite (*replace*) is presented. Some lessons learned are reported. This work is an extension of a conference paper (Chen et al. 2002 ISSTA)						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:	Symbolic execution					
Application domain(s):	Pattern matching					
Application scenarios						Number of MRs
Regular expression matching						4
				Total:		4
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Replace (Siemens suite)	C	563	☐	5542	32	0
			☐			
Total						
Source TCs generation technique:	Test suite					
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
- Finding good MRs requires knowledge of the problem domain, understanding of user requirements, as well as some creativity. - Different MRs demonstrate different fault-detection capabilities. - Different MRs are complementary to one another. - More than one MR should be used for testing programs.						
Challenges						
- Prioritization of MRs.						

B.47 Ding et al SSIRI'10

See legend in page 24 to know the exact meaning of each field.

2010-ding-ssiri						
Publication data						
Authors:	J. Ding and T. Wu and J. Q. Lu and X. Hu					
Title:	Self-Checked Metamorphic Testing of an Image Processing Program					
Publication:	Fourth International Conference on Secure Software Integration and Reliability Improvement					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2010					
DOI/URL:	http://dx.doi.org/10.1109/SSIRI.2010.25					
Pages:	8					
Country:	United States					
Contact:	dingj@ecu.edu					
Summary:						
This paper proposed a combined approach of MT with structural testing (coverage criteria). The authors argue that for some random inputs MRs could still hold remaining faults undetected. To further explore those MR, the authors propose using coverage criteria to detect differences on the execution of source and follow-up test cases revealing faults even when MRs hold. A case study with a cellular image processing program is presented.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:	Structural testing (coverage)					
Application domain(s):	Computer graphics					
Application scenarios					Number of MRs	
Cellular image processing (3D reconstruction of mitochondrion structure)					5	
Total:					5	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Program for 3D cell structure reconstruction	Fortran 90	5600	☐	36	1	0
			☐			
Total		5600			1	
Source TCs generation technique:	Test suite					
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.48 Dong et al ICWIIAT'10

See legend in page 24 to know the exact meaning of each field.

2010-dong-icwiiat						
Publication data						
Authors:	G. Dong and S. Wu and G. Wang and T. Guo and Y. Huang					
Title:	Security Assurance with Metamorphic Testing and Genetic Algorithm					
Publication:	Workshop on Service Intelligence and Engineering					
Pub. Type:	☐ Journal ☐ Conference / Symp. ☒ Workshop ☐ Other:					
Year:	2010					
DOI/URL:	http://dx.doi.org/10.1109/WI-IAT.2010.101					
Pages:	5					
Country:	China					
Contact:	guotao@itsec.gov.cn					
Summary:						
This paper proposes the combination of genetic algorithm and metamorphic testing for the generation of test data. In particular, the authors propose using MRs as part of the fitness function to accelerate the convergence of the search toward the target. Two small case studies with numerical programs are presented. The results suggest that the approach generates more effective test data than pure search-based testing.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:	Search-based testing					
Application domain(s):	Numerical programs					
Application scenarios						Number of MRs
TriSquare: Decides whether 3 integers could form a triangle.						2
Determinant computation						1
					Total:	3
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
TriSquare	Java	30	☐	NR	2	0
Determinant	Java	30	☐	NR	1	0
Total					3	
Source TCs generation technique:	Search-based generation					
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.49 Just and Schweiggert AST'10

See legend in page 24 to know the exact meaning of each field.

2010-just-ast						
Publication data						
Authors:	R. Just and F. Schweiggert					
Title:	Automating Software Tests with Partial Oracles in Integrated Environments					
Publication:	5th Workshop on Automation of Software Test					
Pub. Type:	☐ Journal ☐ Conference / Symp. ☒ Workshop ☐ Other:					
Year:	2010					
DOI/URL:	http://dx.doi.org/10.1145/1808266.1808280					
Pages:	4					
Country:	Germany					
Contact:	rene.just@uni-ulm.de					
Summary:						
This paper presents a case study on the use of MT to test the individual parts of an integrated system for image processing. Four MRs are defined and applied to the open source tool JJ2000 from which 2183 mutants were generated. The results suggest that the MRs used to test part of the systems may not show the same effectiveness when used to test the whole application. Combining MRs is suggested as a way to compensate such variations. A similar case study was presented by the authors in a previous conference paper (Just and Schweiggert 2009 ICSTW).						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Computer graphics					
Application scenarios						Number of MRs
Image preprocessing and decorrelation						4
				Total:		4
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
JJ2000 library	Java	NR	☒	NR	2183	0
			☐			
Total		4396			2183	
Source TCs generation technique:	Random					
Evaluation metrics:	Mutation score					
☐ Available evaluation material						
Lessons learned / guidelines						
- The combination of MRs can significantly increase their effectiveness. - Combining the most effective MRs is nearly as effective as combining all MRs.						
Challenges						

B.50 Kuo et al. IET'10

See legend in page 24 to know the exact meaning of each field.

2010-kuo-iet						
Publication data						
Authors:	F. Kuo and Z. Zhou and J. Ma and G. Zhang					
Title:	Metamorphic testing of decision support systems: a case study					
Publication:	IET Software Journal					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2010					
DOI/URL:	http://dx.doi.org/10.1049/iet-sen.2009.0084					
Pages:	8					
Country:	Australia					
Contact:	zhiquan@uow.edu.au					
Summary:						
This paper presents an approach for the automated detection of faults in decision support systems. In particular, they focus on the so—called Multi--Criteria Group Decision Making (MCGDM), in which decision problems are modelled as a matrix with several dimensions: alternatives, criteria and experts. They also introduced eleven metamorphic relations in natural language, and evaluated their approach using artificial faults in the research tool Decider. The results show that MT is effective for fault detection in decision support systems.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Decision support systems					
Application scenarios						Number of MRs
Multi-criteria group decision making						11
				Total:		11
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Decider (release Sept 2008)	Java	12795	☐	1000	10	1
			☐			
Total		12795		1000	10	1
Source TCs generation technique:	Random					
Evaluation metrics:	Number of test failed test cases for each mutant and MR					
☐ Available evaluation material						
Lessons learned / guidelines						
- Many different MRs can be combined to improve fault-detection effectiveness. - If the initial and follow-up test case are very different, then the chance of violating an MR will be relatively higher.						
Challenges						
- Prioritization of MRs.						

B.51 Liu et al. CSEET'10

See legend in page 24 to know the exact meaning of each field.

2010-liu-cseet						
Publication data						
Authors:	H. Liu and F. Kuo and T. Y. Chen					
Title:	Teaching an End-User Testing Methodology					
Publication:	23rd Conference on Software Engineering Education and Training					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2010					
DOI/URL:	http://dx.doi.org/10.1109/CSEET.2010.28					
Pages:	8					
Country:	Australia					
Contact:	hliu@swin.edu.au					
Summary:						
The paper reports a 3-year experience in teaching MT to various groups of students at Swinburne University of Technology (Australia). The authors explain the teaching approach followed and the main results including a number of lessons learned. The main conclusion is that MT is a suitable technique for end-user engineering.						
Contribution						
☐ New technique / method ☐ Survey / overview ☒ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):						
Application scenarios					Number of MRs	
Total:						
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
- Most students understood the concepts and principles of MT and learned how to use MT. - Most students were able to identify correct MRs based on the authors' guidance. - Different students identified different MRs that target different faults. - The majority of students were able to automate MT without the supporting tool. - The test drivers developed by different students have different failure-detection effectiveness.						
Challenges						

B.52 Lu et al. UATC'10

See legend in page 24 to know the exact meaning of each field.

2010-lu-uatc						
Publication data						
Authors:	X. Lu and Y. Dong and C. Luo					
Title:	Testing of Component-based Software: a Metamorphic Testing Methodology					
Publication:	Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2010					
DOI/URL:	http://dx.doi.org/10.1109/UIC-ATC.2010.75					
Pages:	5					
Country:	China					
Contact:	luxl73@nwu.edu.cn					
Summary:						
This paper proposes a MT-based methodology for testing component-based applications. First, the basic steps of the methodology are presented. Then, they are illustrated in a trivial scenario. No experimental evaluation is reported.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Component-based software					
Application scenarios						Number of MRs
Foreign exchange component						3
					Total:	3
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.53 Segura et al. ICST'10

See legend in page 24 to know the exact meaning of each field.

2010-segura-icst						
Publication data						
Authors:	S. Segura and R. M. Hierons and D. Benavides and A. Ruiz-Cortés					
Title:	Automated Test Data Generation on the Analyses of Feature Models: A Metamorphic Testing Approach					
Publication:	Third International Conference on Software Testing, Verification and Validation					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2010					
DOI/URL:	http://dx.doi.org/10.1109/ICST.2010.20					
Pages:	10					
Country:	Spain					
Contact:	sergiosegura@us.es					
Summary:						
This paper proposes the use of MT to detect faults in feature model analysis tools. The authors present a number of MRs and a test data generator relying on them. In contrast to related works on MT, the authors do not propose checking the results of source and follow-up test cases. Instead, MRs are used to compute the actual output of follow-up test cases. This enables the iterative application of MRs generating non-trivial input feature models and their corresponding (potentially huge) set of products. The approach is evaluated using mutation testing in three open-source feature model analysis tools. Also, two defects were found in the tool FaMa.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Automated analysis of feature models					
Application scenarios						Number of MRs
Automated analysis of feature models (6 operations)						6
				Total:		6
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Sat4jReasoner v0.9.2	Java	743	☒	NR	188	0
JavaBDDReasoner v0.9.2	Java	625	☒	NR	237	0
JaCoPReasoner v0.8.3	Java	686	☒	NR	136	0
FaMa v1.0.0 alpha	Java	NR	☒	NR	0	2
Total					561	2
Source TCs generation technique:	Random					
Evaluation metrics:						
☒ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.54 Segura et al. IST'10

See legend in page 24 to know the exact meaning of each field.

2010-segura-ist						
Publication data						
Authors:	S. Segura and R. M. Hierons and D. Benavides and A. Ruiz-Cortés					
Title:	Automated metamorphic testing on the analyses of feature models					
Publication:	Information and Software Technology Journal					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2010					
DOI/URL:	http://dx.doi.org/10.1016/j.infsof.2010.11.002					
Pages:	14					
Country:	Spain					
Contact:	sergiosegura@us.es					
Summary: This paper proposes the use of MT to detect faults in feature model analysis tools. The authors present a number of MRs and a test data generator relying on them. In contrast to related works on MT, the authors do not propose checking the results of source and follow-up test cases. Instead, MRs are used to compute the actual output of follow-up test cases. This enables the iterative application of MRs generating non-trivial input feature models and their corresponding (potentially huge) set of products. The approach is evaluated using mutation testing in three open-source feature model analysis tools. Also, two defects were found in the tool FaMa and another two in the tool SPLAR (analysis engine of SPLOT). This work is an extension of a conference paper (Segura et al. 2010 ICST).						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Automated analysis of feature models					
Application scenarios						Number of MRs
Automated analysis of feature models (7 operations)						6
				Total:		6
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Sat4jReasoner v0.9.2	Java	743	☒		188	
JavaBDDReasoner v0.9.2	Java	625	☒		237	
JaCoPReasoner v0.8.3	Java	686	☒		136	
FaMa v1.0.0 alpha	Java		☒			2
SPLAR Feb 2010			☒			2
Total					561	4
Source TCs generation technique:	Random					
Evaluation metrics:	Mutation score					
☒ Available evaluation material						
Lessons learned / guidelines						
MT produces better results when combined with other strategies (other than random testing) for the selection of source test cases. The improvement, however, was noticed in terms of detection time but not in terms of fault detection capability.						
Challenges						

B.55 Sim et al. ICISE'10

See legend in page 24 to know the exact meaning of each field.

2010-sim-icise						
Publication data						
Authors:	K. Y. Sim and C. S. Low and F. Kuo					
Title:	Detecting Faults in Technical Indicator Computations for Financial Market Analysis					
Publication:	2nd International Conference on Information Science and Engineering					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2010					
DOI/URL:	http://dx.doi.org/10.1109/ICISE.2010.5689221					
Pages:	6					
Country:	Malaysia					
Contact:	ksim@swinburne.edu.my					
Summary:						
The paper presents a MT approach for the detection of faults in financial software. The authors first present several technical indicators and several MRs for each of them. Then, they generate several mutants of the commercial tool Metatrader and check how many of them are detected by the proposed MRs. Tests are integrated in the tool in a self-testing strategy. Source and follow-up test cases are obtained from the run-time input price data received at different period of times. Results suggest that MT is effective in detecting faults in financial software suffering from the oracle problem.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Financial software					
Application scenarios						Number of MRs
Simple Moving Averages (SMA)						2
Smoothed Moving Averages (SMMA)						4
Relative Strength Index (RSI)						2
					Total:	8
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
MetaTrader 4 client terminal	NR	NR	☒	2000	8	0
			☐			
Total					8	
Source TCs generation technique:	Test suite (run-time price data)					
Evaluation metrics:	Number of mutants killed by each MR					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.56 Tao et al. APSEC'10

See legend in page 24 to know the exact meaning of each field.

2010-tao-apsec						
Publication data						
Authors:	Q. Tao and W. Wu and C. Zhao and W. Shen					
Title:	An Automatic Testing Approach for Compiler Based on Metamorphic Testing Technique					
Publication:	17th Asia Pacific Software Engineering Conference					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2010					
DOI/URL:	http://dx.doi.org/10.1109/APSEC.2010.39					
Pages:	10					
Country:	China					
Contact:	qiuming@iscas.ac.cn					
Summary:						
The paper presents and MT-based approach for testing compilers. First, the author proposes a so-called equivalence preservation metamorphic relation. Then, three different strategies for the generation of input equivalent source programs are presented. The approach is implemented in a tool named Mettoc. Finally, a case study with several open source C compilers and mutation is presented. Among other results, two real defects were detected.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☒ Tool						
Combination with other techniques:						
Application domain(s):	Compilers					
Application scenarios						Number of MRs
Source code compilation						3
					Total:	3
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
GCC v4.4.2	C	NR	☒	2700	621	0
GCC v4.4.3	C	NR	☒	NR	0	1
PCC v0.9.9	C	NR	☒	NR	0	0
TCC v0.9.25	C	NR	☒	NR	0	0
UCC v1.6	C	NR	☒	NR	0	1
Total					621	2
Source TCs generation technique:	Random					
Evaluation metrics:	Mutation score					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.57 Xie et al. JSS'10

See legend in page 24 to know the exact meaning of each field.

2010-xie-jss						
Publication data						
Authors:	X. Xie and J. W. K. Ho and C. Murphy and G. Kaiser and B. Xu and T. Y. Chen					
Title:	Testing and validating machine learning classifiers by metamorphic testing					
Publication:	The Journal of Systems and Software					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2010					
DOI/URL:	http://dx.doi.org/10.1016/j.jss.2010.11.920					
Pages:	15					
Country:	Australia					
Contact:	xxie@groupwise.swin.edu.au					
Summary:						
These authors propose using MT for the detection of faults in supervised classifiers. They argue that MT can be helpful for both validation and verification. Validation is used to find out whether the algorithm is appropriate for the problem. Verification is used to detect fault in the algorithms. Two specific algorithms are studied: K-Nearest Neighbours (KNN) and Naïve Bayes Classifier (NBC). As a first step, 11 MRs are proposed and applied to implementations of the algorithms in the tool Weka. The results reveal several defects in the implementation of NBC. A further validation is reported using mutation analysis and cross-validation. This work is an extension of a previous conference paper (Xie et al. 2009 QSIC).						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Machine learning					
Application scenarios						Number of MRs
K-Nearest neighbours						11
Naïve Bayes Classifier						9
					Total:	20
Evaluation						
Program	Language	Size	Rea	STCs	Mutants	Faults
Weka 3.5.7	Java	16.4M	☒	300	50	3
			☐			
Total				300	50	3
Source TCs generation technique:	Random					
Evaluation metrics:	Percentage of test cases violating a MR.					
☐ Available evaluation material						
Lessons learned / guidelines						
- Equality MRs are preferred because an equality expression is tighter than a non-equality one. - Different MRs have different performance in detecting program faults. - Combination of MRs may lead to better failure-detection capabilities.						
Challenges						

B.58 Yoo ICSTW'10

See legend in page 24 to know the exact meaning of each field.

2010-yoo-icstw						
Publication data						
Authors:	S. Yoo					
Title:	Metamorphic Testing of Stochastic Optimisation					
Publication:	3rd International Workshop on Search-Based					
Pub. Type:	☐ Journal ☐ Conference / Symp. ☒ Workshop ☐ Other:					
Year:	2010					
DOI/URL:	http://dx.doi.org/10.1109/ICSTW.2010.26					
Pages:	10					
Country:	United kingdom					
Contact:	Shin.Yoo@kcl.ac.uk					
Summary:						
This paper proposes a MT-based approach for stochastic optimization algorithms. More specifically, the authors apply the Statistical Metamorphic Testing (SMT) approach presented by Guderlei and Mayer (QSIC 2007) to the context of metaheuristics. Since metaheuristic algorithms are by nature stochastic, the authors propose to compare the output of different executions using statistical hypothesis testing. A case study with a simulated annealing algorithm and the next release problem is presented. The results show that SMT can be effective for certain class of faults in optimization algorithms. It also shows that the effectiveness of SMT not only depends on the algorithm and the fault but also son the problem instance used for the test.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Optimization algorithms					
Application scenarios						Number of MRs
Simulated Annealing – Next Release Problem						1
				Total:		1
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Simulated Annealing	Java	25	☐	NR	86	0
			☐			
Total		25			86	
Source TCs generation technique:	Test suite and random generation					
Evaluation metrics:	Mutation score					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.59 Zhou et al. STVR'10

See legend in page 24 to know the exact meaning of each field.

2010-zhou-stvr						
Publication data						
Authors:	Z. Zhou and S. Zhang and M. Hagenbuchner and T. H. Tse and F. Kuo and T. Y. Chen					
Title:	Automated functional testing of online search services					
Publication:	Software Testing, Verification and Reliability Journal					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2010					
DOI/URL:	http://dx.doi.org/10.1002/stvr.437					
Pages:	23					
Country:	Australia					
Contact:	zhiquan@uow.edu.au					
Summary: This article proposes using MT for the detection of inconsistencies in online search services. Several MRs are proposed and used in a number of experiments with three Web search engines: Google, Yahoo and Live Search. The results show that MT effectively detects inconsistencies in the searches in terms of both returned content and ranking quality.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Online search services					
Application scenarios						Number of MRs
Online search engine						7
				Total:		7
Evaluation						
Program	Language	Size	Real	STCs	Mutant	Faults
Google			☒	>1000		
Yahoo			☒	>1000		
Live Search			☒	>1000		
Total				>3000		Not explicitly reported
Source TCs generation technique:	Random					
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.60 Asrafi et al. SSIRI'11

See legend in page 24 to know the exact meaning of each field.

2011-asrafi-ssiri						
Publication data						
Authors:	M. Asrafi and H. Liu and F. Kuo					
Title:	On Testing Effectiveness of Metamorphic Relations: A Case Study					
Publication:	Fifth International Conference on Secure Software Integration and Reliability Improvement					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2011					
DOI/URL:	http://dx.doi.org/10.1109/SSIRI.2011.21					
Pages:	10					
Country:	Australia					
Contact:	hliu@swin.edu.au					
Summary: This paper presents a case study to explore the correlation between the execution behaviour and the fault-effectiveness of MRs. Two sample programs are used as the subjects of the case study. First, the programs are run with thousands of test cases (and associated follow-up test cases) measuring the line and branch coverage. Then, the fault-detection effectiveness of the proposed MRs is measured using mutation analysis. Finally, the correlation between coverage and fault-detection effectiveness is statistically analysed. The authors conclude that execution behaviour is, in general, a good indicator of the fault effectiveness of MRs. However, they also point out that other aspects must be considered as the program structure. MRs with low coverage could still be helpful if the code coverage is not exercised by other MRs.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☐ Case study / application ☒ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Conflict detection, optimization					
Application scenarios						Number of MRs
Onboard aircraft conflict detection and resolution						14
Knapsack						10
					Total:	24
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
TCAS	C	173	☐	10000	422	0
Knapsack	Java	180	☐	10000	100	0
Total				20000	522	0
Source TCs generation technique:	Random					
Evaluation metrics:	MR Probability (MRP) and Fault Detection Probability within a Range (FDPR)					
☐ Available evaluation material						
Lessons learned / guidelines						
- There exists a correlation between the code coverage and the fault-detection effectiveness of MRs. - Other factors, apart from the code coverage, must be considered when designing MRs e.g. program structure. - There can be some situations where some program segments are covered by some MRs with low coverage, but not by those with high coverage.						
Challenges						

B.61 Barus et al. SET'11

See legend in page 24 to know the exact meaning of each field.

2011-barus-set						
Publication data						
Authors:	A. C. Barus and T. Y. Chen and D. Grant and F. Kuo and M. F. Lau					
Title:	Testing of Heuristic Methods: A Case Study of Greedy Algorithm					
Publication:	Third IFIP TC 2 Central and East European conference on Software engineering techniques					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2011					
DOI/URL:	http://dx.doi.org/10.1007/978-3-642-22386-0_19					
Pages:	15					
Country:	Australia					
Contact:	abarus@ict.swin.edu.au					
Summary:						
This paper presents a case study on the use of MT for the detection of faults in a greedy algorithm for solving the Key-Lock Problem (KLP). Nine MRs are identified. An experiment is conducted to measure the fault-detection effectiveness of the proposed MRs using five seeded-faults.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Optimization algorithms					
Application scenarios						Number of MRs
Key-Lock Problem (KLP)						9
				Total:		9
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
KLP	Java	35	☐	100	5	0
			☐			
Total				100	5	
Source TCs generation technique:	Random					
Evaluation metrics:	Percentage of test cases that detected a mutant M using metamorphic relation MR, percentage of test cases that detected any mutant.					
☐ Available evaluation material						
Lessons learned / guidelines						
- Different failures can be revealed by different MRs. - Some MRs can reveal more failures than others.						
Challenges						
- Selection and prioritization of MRs						

B.62 Batra and Sengupta ISTM'11

See legend in page 24 to know the exact meaning of each field.

2011-batra-istm						
Publication data						
Authors:	G. Batra and J. Sengupta					
Title:	An Efficient Metamorphic Testing Technique Using Genetic Algorithm					
Publication:	5th International Conference on Information Intelligence, Systems, Technology and Management					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2011					
DOI/URL:	http://dx.doi.org/10.1007/978-3-642-19423-8_19					
Pages:	9					
Country:	India					
Contact:	gdeep.pbi@gmail.com					
Summary:						
The paper proposes using a genetic algorithm for the optimized selection of source test cases for metamorphic testing, named "genetically augmented metamorphic testing". More specifically, the authors propose using the traversed paths in the SUT to guide the search toward test cases that exercise the most critical paths in the program. This is therefore a white-box approach. A small experiment with a C program for determining the type of a triangle and 4 mutants is reported.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:	Search-based testing					
Application domain(s):	Numerical program					
Application scenarios						Number of MRs
Triangle type determination program						5
				Total:		5
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Tritype	C	32	☐	NR	4	0
			☐			
Total		32			4	
Source TCs generation technique:	Search-based generation					
Evaluation metrics:	Mutation score and fault detection ratio					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.63 Castro-Cabrera and Medina-Bulo ICEB'11

See legend in page 24 to know the exact meaning of each field.

2011-castro-iceb						
Publication data						
Authors:	C. Castro-Cabrera and I. Medina-Bulo					
Title:	An approach to metamorphic testing for WS-BPEL compositions					
Publication:	International Conference on e-Business					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2011					
DOI/URL:	http://dx.doi.org/10.5220/0003611401370142					
Pages:	6					
Country:	Spain					
Contact:	maricarmen.decastro@uca.es					
Summary:						
This short paper describes a theoretical approach for metamorphic testing of WS-BPEL compositions. The authors explain the main steps of the future system and detail how it will be supported in the work of previous authors. An illustrative example is presented with a Loan Approval Composition.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Web service composition					
Application scenarios						Number of MRs
					Total:	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.64 Ding et al. AST'11

See legend in page 24 to know the exact meaning of each field.

2011-ding-ast						
Publication data						
Authors:	J. Ding and T. Wu and D. Xu and J. Q. Lu and X. Hu					
Title:	Metamorphic Testing of a Monte Carlo Modeling Program					
Publication:	6th International Workshop on Automation of Software Test					
Pub. Type:	☐ Journal ☐ Conference / Symp. ☒ Workshop ☐ Other:					
Year:	2011					
DOI/URL:	http://dx.doi.org/10.1145/1982595.1982597					
Pages:	7					
Country:	United States					
Contact:	dingj@ecu.edu					
Summary:						
The paper presents a MT-based approach for testing a Monte Carlo program for the simulation of photon propagation. In particular, the authors propose using a self-checked metamorphic testing approach (Ding et al. 2010 SSIRI). In this approach, MT is extended using code coverage criteria to evaluate the quality of the proposed MRs. Five MRs are presented and used to test a Monte Carlo program written in Fortran 90. Authors conclude that testing coverage information effectively guide the selection of MRs and the creation of test cases.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:	Structural testing (coverage)					
Application domain(s):	Simulation (stochastic techniques)					
Application scenarios						Number of MRs
Monte Carlo program of photon transportation						5
				Total:		5
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Monte Carlo program	Fortran 90	1600	☐	NR	0	0
			☐			
Total		1600				
Source TCs generation technique:	Test suite					
Evaluation metrics:	Code coverage					
☐ Available evaluation material						
Lessons learned / guidelines						
- Structural testing is helpful to guide the selection of MRs and test cases.						
Challenges						

B.65 Jing et al. JE'11

See legend in page 24 to know the exact meaning of each field.

2011-jing-je						
Publication data						
Authors:	J. Zhang and X. Hu and B. Zhang					
Title:	An evaluation approach for the program of association rules algorithm based on metamorphic relations					
Publication:	Journal of Electronics (China)					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2011					
DOI/URL:	http://dx.doi.org/10.1007/s11767-012-0743-9					
Pages:	9					
Country:	China					
Contact:	Zhangjing@hfut.edu.cn					
Summary: This article presents a case study on the use of MT in association rules programs in the context of data mining. Seven MRs are presented and used to test one of the algorithm integrated in the machine learning tool Weka.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Machine learning					
Application scenarios						Number of MRs
Association rules algorithm						7
				Total:		7
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Weka	Java	NR	☒	124		
			☐			
Total						
Source TCs generation technique:	Test suite (contact-lenses dataset) and random test cases					
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.66 Just and Schweiggert SQJ'11

See legend in page 24 to know the exact meaning of each field.

2011-just-sqj						
Publication data						
Authors:	R. Just and F. Schweiggert					
Title:	Automating unit and integration testing with partial oracles					
Publication:	Software Quality Control Journal					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2011					
DOI/URL:	http://dx.doi.org/10.1007/s11219-011-9151-x					
Pages:	17					
Country:	Germany					
Contact:	rene.just@uni-ulm.de					
Summary:						
This paper presents a case study on the use of MT to test the individual parts of an integrated system for image processing. Seven MRs are defined and applied to the open source tool JJ2000 from which 2183 mutants were generated. The results suggest that the MRs used to test part of the systems may not show the same effectiveness when used to test the whole application. Combining MRs is suggested as a way to compensate such variations. This work is an extension of a previous workshop paper (Just and Schweiggert 2010 AST).						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☒ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Computer graphics					
Application scenarios					Number of MRs	
Image preprocessing and decorrelation					7	
				Total:	7	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
JJ2000 library	Java	4396	☒	NR	2183	0
			☐			
Total		4396			2183	
Source TCs generation technique:	Random					
Evaluation metrics:	Mutation score					
☐ Available evaluation material						
Lessons learned / guidelines						
- The combination of MRs can significantly increase their effectiveness. - Combining the most effective MRs is nearly as effective as combining all MRs. - When constructing MRs, it is advisable to exploit constraints like equivalence relations in conjunction with properties such as commutativity, distributive or associativity. - For efficiency reasons, the combination of necessary conditions should be implemented within a single MR even although the complexity is increased. - The partial oracles derived from the characteristics of the integrated (sub)systems may be less effective than partial oracles for the individual parts of the system.						
Challenges						

B.67 Kuo et al. LCN'11

See legend in page 24 to know the exact meaning of each field.

2011-kuo-lcn						
Publication data						
Authors:	F. Kuo and T. Y. Chen and W. K. Tam					
Title:	Testing Embedded Software by Metamorphic Testing: a Wireless Metering System Case Study					
Publication:	36th Annual IEEE Conference on Local Computer Networks					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2011					
DOI/URL:	10.1109/LCN.2011.6115306					
Pages:	4					
Country:	Australia					
Contact:	dkuo@ict.swin.edu.au					
Summary: This paper proposes using MT for the detection of faults in embedded software. A case study is reported on the use of MT in a wireless metering system. One MR was identified and used to test the meter reading function of a commercial device from the electric industry. Two real defects were uncovered.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Embedded software					
Application scenarios						Number of MRs
Wireless metering system						1
				Total:		1
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
RF-Soft	C	NR	☒	NR	NR	2
			☐			
Total						2
Source TCs generation technique:	NR					
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						
- Identification of MRs.						

B.68 Kuo et al. SAC'11

See legend in page 24 to know the exact meaning of each field.

2011-kuo-sac						
Publication data						
Authors:	F. Kuo and S. Liu and T. Y. Chen					
Title:	Testing a Binary Space Partitioning Algorithm with Metamorphic Testing					
Publication:	2011 ACM Symposium on Applied Computing					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2011					
DOI/URL:	http://dx.doi.org/10.1145/1982185.1982502					
Pages:	8					
Country:	Australia					
Contact:	dkuo@groupwise.swin.edu.au					
Summary:						
The paper proposes using MT for detection of faults in a binary space partitioning algorithm. Five MRs are presented and used to test an implementation of the surface visibility problem using Binary Space Partitioning (BSP) tree. One real fault and ten mutants were effectively detected.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Computer graphics					
Application scenarios						Number of MRs
Surface visibility using Binary Space Partitioning (BSP) tree						5
					Total:	5
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
BSP-treeVS	C	NR	☐	5000	10	1
			☐			
Total				5000	10	1
Source TCs generation technique:	Random					
Evaluation metrics:	Number of test cases detecting a mutant M using metamorphic relation R					
☐ Available evaluation material						
Lessons learned / guidelines						
- Different faults are sensitive to different MRs.						
Challenges						

B.69 Murphy et al. SEHC'11

See legend in page 24 to know the exact meaning of each field.

2011-murphy-sehc						
Publication data						
Authors:	C. Murphy and M. S. Raunak and A. King and S. Chen and C. Imbriano and G. Kaiser and I. Lee and O. Sokolsky and L. Clarke and L. Osterweil					
Title:	On Effective Testing of Health Care Simulation Software					
Publication:	3rd Workshop on Software Engineering in Health Care					
Pub. Type:	☐ Journal ☐ Conference / Symp. ☒ Workshop ☐ Other:					
Year:	2011					
DOI/URL:	http://dx.doi.org/10.1145/1987993.1988003					
Pages:	8					
Country:	United States					
Contact:	cdmurphy@cis.upenn.edu					
Summary: This paper proposes using MT for the detection of faults in simulation software. Some guidelines for the design of MRs are reported. To show the feasibility of the approach, a case study with two health care simulators (JSIM and GCS) and mutation analysis is presented.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Simulation software					
Application scenarios					Number of MRs	
Discrete event simulation					3[1]	
Glycemic control simulation					3	
				Total:	6	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
JSim	Java	NR	☐	NR	25	0
GCS	MATLAB	NR	☐	NR	724	0
Total					749	0
Source TCs generation technique:	Test suite (Emergency department model)					
Evaluation metrics:	Mutation score					
☐ Available evaluation material						
Lessons learned / guidelines						
- MRs should consider: 1) Properties shared by all the applications in a given domain, 2) properties specific to the algorithm under test, and 3) properties applicable only to a given input.						
Challenges						

[1] The number of MRs is not explicitly specified in the paper.

B.70 Sun et al. ICWS'11

See legend in page 24 to know the exact meaning of each field.

2011-sun-icws						
Publication data						
Authors:	C. Sun and G. Wang and B. Mu and H. Liu and Z. Wang and T. Y. Chen					
Title:	Metamorphic Testing for Web Services: Framework and a Case Study					
Publication:	International Conference on Web Services					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2011					
DOI/URL:	http://dx.doi.org/10.1109/ICWS.2011.65					
Pages:	8					
Country:	China					
Contact:	casun@ustb.edu.cn					
Summary:						
This paper presents a framework for the application of MT on web services. In particular, the authors propose several steps and theoretical tools (e.g. "test case generator") for the application of MT in the context of SOA. MRs are expected to be provided by the tester. A small case study is presented using an electronic payment web service and mutation analysis.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Service-oriented applications					
Application scenarios					Number of MRs	
Electronic payment					6	
				Total:	6	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
ATM Web Service	Java	136	☐	50-200	129	0
			☐			
Total		136		50-200	129	
Source TCs generation technique:	Random					
Evaluation metrics:	Mutation score and fault discovery rate.					
☐ Available evaluation material						
Lessons learned / guidelines						
- Each MR has a varying sensitivity to different mutants.						
Challenges						

B.71 Xie et al. QSIC'11

See legend in page 24 to know the exact meaning of each field.

2011-xie-qsic						
Publication data						
Authors:	X. Xie and W. E. Wong and T. Y. Chen and B. Xu					
Title:	Spectrum-Based Fault Localization: Testing Oracles Are No Longer Mandatory					
Publication:	11th International Conference On Quality Software					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2011					
DOI/URL:	http://dx.doi.org/10.1109/QSIC.2011.20					
Pages:	10					
Country:	Australia					
Contact:	xxie@groupwise.swin.edu.au					
Summary:						
The paper proposes using MT to extend the applicability of spectrum-based fault localization to programs without oracles. In particular, the authors propose the concept of "mice", based on the integration of MRs and slices. A case study with 9 programs and mutation analysis is presented. The results suggest that the approach is applicable offering a fault detection capability similar to the conventional spectrum-based fault localization techniques.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:	Spectrum-based fault localization					
Application domain(s):	Pattern matching, lexical analyser, priority scheduler, altitude separation, information measure, bioinformatics.					
Application scenarios					Number of MRs	
Grep – pattern matching					3[1]	
Short sequence mapping (bioinformatics)					3	
Lexical analyser					3	
Priority scheduler					3	
Altitude separation					3	
Information measure					3	
String matching					3	
				Total:	21	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Fault
print_tokens	C	342	☐	4130	77 max[2]	0
print_tokens2	C	355	☐	4115	79 max	0
replace	C	512	☐	5542	144 max	0
schedule	C	292	☐	2650	98 max	0
schedule2	C	262	☐	2710	94 max	0
tcas	C	135	☐	1608	69 max	0
tot_info	C	273	☐	1052	76 max	0
seqMap v1.0.8	C++	1783	☒	300	97 max	0
grep 1.2	C	7309	☒	10069	146 max	0
Total		11270		32176	880 max	0
Source TCs generation technique:	Test suite and random testing					

Evaluation metrics:	EXAM score (percentage of executable statements that have to be examined until the first statement containing the bug is reached)
☐ Available evaluation material	
Lessons learned / guidelines	
Challenges	

[1] These are the only MRs explicitly presented in the paper.
[2] The exact number of mutants is not reported. This is the maximum number of mutants according to the number of mutants used for each MR on each program.

B.72 Castro-Cabrera and Medina-Bulo EBT'12

See legend in page 24 to know the exact meaning of each field.

2012-castro-ebt						
Publication data						
Authors:	C. Castro-Cabrera and I. Medina-Bulo					
Title:	Application of Metamorphic Testing to a Case Study in Web Services Compositions					
Publication:	E-Business and Telecommunications					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other: Book ch.					
Year:	2012					
DOI/URL:	http://dx.doi.org/10.1007/978-3-642-35755-8_13					
Pages:	14					
Country:	Spain					
Contact:	maricarmen.decastro@uca.es					
Summary:						
The paper presents a MT-based approach for WS-BPEL Web service compositions. A small case study with a loan approval composition and three MRs is presented to show the feasibility of the approach. This paper is an extension of a previous work (Castro et al. 2011 ICEB)						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Web service compositions					
Application scenarios					Number of MRs	
Loan approval web service composition					3	
				Total:	3	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.73 Chen et al. ISSDM'12

See legend in page 24 to know the exact meaning of each field.

2012-chen-issdm						
Publication data						
Authors:	L. Chen and L. Cai and J. Liu and Z. Liu and S. Wei and P. Liu					
Title:	An optimized method for generating cases of metamorphic testing					
Publication:	6th International Conference on New Trends in Information Science and Service Science and Data Mining					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2012					
DOI/URL:	http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6528673					
Pages:	5					
Country:	China					
Contact:	cll1128@163.com					
Summary:						
This paper presents a method to obtain a minimum set of effective source test cases for MT. In particular, the authors propose an algorithm for the generation of test cases satisfying the so-called ECCEM (Equivalence-Class Coverage for Every Metamorphic Relation) criterion. A small case study is presented using mutation testing. The author conclude that selecting a source test case from each equivalence class lead to test cases with both a high utilization rate and high failure-detection capability. This work is highly inspired in the work of Dong et al. (Dong et al. 2007 QSIC)						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:	Equivalence class partitioning					
Application domain(s):	Numerical programs					
Application scenarios						Number of MRs
TriSquare: Check whether 3 positive real numbers could construct a triangle						7
					Total:	7
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
TriSquare	Java	30	☐	NR	4	0
			☐			
Total		30			4	0
Source TCs generation technique:	Test suite (Equivalent class partitioning)					
Evaluation metrics:	Mutation score and test case rate of utilization.					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.74 Chen et al. QSIC'12

See legend in page 24 to know the exact meaning of each field.

2012-chen-qsic						
Publication data						
Authors:	T. Y. Chen and F. Kuo and D. Towey and Z. Zhou					
Title:	Metamorphic Testing: Applications and Integration with Other Methods					
Publication:	International Workshop on Embedded System Software Development and Quality Assurance					
Pub. Type:	☐ Journal ☐ Conference / Symp. ☒ Workshop ☐ Other:					
Year:	2012					
DOI/URL:	http://dx.doi.org/10.1109/QSIC.2012.21					
Pages:	4					
Country:	Australia					
Contact:	tychen@groupwise.swin.edu.au					
Summary: This tutorial synopsis presents an introduction to MT outlining some of its applications and the integration with other testing techniques.						
Contribution						
☐ New technique / method ☒ Survey / overview ☐ Empirical study ☐ Other: ☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Graphs, optimization program, online search services, wireless embedded software.					
Application scenarios					Number of MRs	
Shortest path					2	
Quadratic Assignment Problem (QAP)					3	
				Total:	5	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
- MT can also be regarded as a test case generation strategy because follow-up test cases can be generated from source test cases by referring to MRs.						
Challenges						

B.75 Gagandeep and Singh CCIS'12

See legend in page 24 to know the exact meaning of each field.

2012-gagandeep-ccis						
Publication data						
Authors:	G. Batra and G. Singh					
Title:	An Automated Metamorphic Testing Technique for Designing Effective Metamorphic Relations					
Publication:	5th International Conference on Communications in Computer and Information Science					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2012					
DOI/URL:	http://dx.doi.org/10.1007/978-3-642-32129-0_20					
Pages:	12					
Country:	India					
Contact:	gdeep.pbi@gmail.com					
Summary:						
This paper presents a case study on the use of MT in a Banking system research program. First, the authors describe how MRs (11 in total) can be derived from the specification of the system. Then, MRs are implemented and executed revealing several faults.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Numerical program					
Application scenarios						Number of MRs
Banking system (deposit module, withdraw module, loan module, fixed deposit module)						11
				Total:		11
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Banking system	C#	NR	☐	NR	0	3
			☐			
Total						
Source TCs generation technique:	Test suite					
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.76 Liu et al. QSIC'12

See legend in page 24 to know the exact meaning of each field.

2012-liu-qsic						
Publication data						
Authors:	H. Liu and X. Liu and T. Y. Chen					
Title:	A New Method for Constructing Metamorphic Relations					
Publication:	12th International Conference on Quality Software					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2012					
DOI/URL:	http://dx.doi.org/10.1109/QSIC.2012.10					
Pages:	10					
Country:	Australia					
Contact:	hliu@swin.edu.au					
Summary:						
This paper presents a new method named *composition of metamorphic relations*. The approach addresses the problem of creating MRs by combining existing ones. The work includes some theoretical definitions and a case study to compare the failure-detection effectiveness of individual MRs and composite MRs. A bioinformatics programs and 11 mutants are used. The results suggest that composite MRs normally has higher (or at least similar) failure-detection capability than each component MR. Also, composite MRs improve the cost-effectiveness of classic MT since it involves a fewer test executions. A similar idea was earlier presented by Dong et al (see 2008-dong-jsu).						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Bioinformatics					
Application scenarios					Number of MRs	
Phylogenetic program					7	
				Total:	7	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Dnapars	NR	NR	☐	500	11	0
			☐			
Total				500	11	0
Source TCs generation technique:	Random					
Evaluation metrics:	Number of test cases that detect a mutant M using metamorphic relation R. Ratio between the number of detected failures and the number of executed test cases.					
☐ Available evaluation material						
Lessons learned / guidelines						
- The composition of k MRs can produce a composite MR that normally has higher (or at least similar) failure-detection capability than each component MR. - Certain MRs (those that involves a weak output relation) may reduce the failure-detection effectiveness of composite MRs. - The cost-effectiveness of composite MRs (where each individual MR is used only once) is normally higher than that of the individual MRs.						
Challenges						
- Identify the MRs that cannot be combined, e.g. MR_{75} in the paper.						

B.77 Pullum and Ozmen BIOMEDCOM'12

See legend in page 24 to know the exact meaning of each field.

2012-pullum-biomedcom						
Publication data						
Authors:	L. L. Pullum and O. Ozmen					
Title:	Early Results from Metamorphic Testing of Epidemiological Models					
Publication:	Workshop on Verification and Validation of Epidemiological Models					
Pub. Type:	☐ Journal ☐ Conference / Symp. ☒ Workshop ☐ Other:					
Year:	2012					
DOI/URL:	http://dx.doi.org/10.1109/BioMedCom.2012.17					
Pages:	6					
Country:	United States					
Contact:	pullumll@ornl.gov					
Summary:						
The paper proposes using MT for the detection of faults in predictive models for disease spread. A case study on the detection of faults in an Agent-Based Model (ABM) of the 1918 Spanish flu is presented. Fourteen MRs were identified and used for testing. This work is closely related to the work of Ramanathan et al. (Ramanathan et al. 2012 BIOMEDCOM).						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Disease spread					
Application scenarios						Number of MRs
Equation-Based Model (EBM)						14
Agent-Based Model (ABM)						
					Total:	14
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
ABM of the 1918 Spanish flu[1] (SIIR model)			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

[1] A prediction model was tested, not a program.

B.78 Ramanathan et al. BIOMEDCOM'12

See legend in page 24 to know the exact meaning of each field.

2012-ramanathan-biomedcom						
Publication data						
Authors:	A. Ramanathan and C. A. Steed and L. L. Pullum					
Title:	Verification of Compartmental Epidemiological Models using Metamorphic Testing, Model Checking and Visual Analytics					
Publication:	Workshop on Verification and Validation of Epidemiological Models					
Pub. Type:	☐ Journal ☐ Conference / Symp. ☒ Workshop ☐ Other:					
Year:	2012					
DOI/URL:	http://dx.doi.org/10.1109/BioMedCom.2012.18					
Pages:	6					
Country:	United States					
Contact:	ramanathana@ornl.gov					
Summary: This paper proposes the use of several techniques, including MT, for the verification of compartmental epidemiological models. The authors introduce epidemiological models and explain how MT could be helpful for the detection of certain faults in the implementation of SIR/SEIR models. A few MRs are introduced. Authors also explain how visualization tools and model checking could be used for the detection of faults in that type of models.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Disease spread					
Application scenarios						Number of MRs
SIR/SEIR epidemiological models						3
				Total:		3
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
SIR/SEIR models[1]			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

[1] A prediction model was tested, not a program.

B.79 Sun et al. IJWSR'12

See legend in page 24 to know the exact meaning of each field.

2012-sun-jwsr						
Publication data						
Authors:	C. Sun and G. Wang and B. Mu and H. Liu and Z. Wang and T. Y. Chen					
Title:	A Metamorphic Relation-Based Approach to Testing Web Services Without Oracles					
Publication:	International Journal of Web Services Research (IJWSR)					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2012					
DOI/URL:	http://dx.doi.org/10.4018/jwsr.2012010103					
Pages:	25					
Country:	China					
Contact:						
Summary:						
This paper presents a metamorphic testing framework for SOAP web services. The author propose to manually derive metamorphic relations from the WSDL description of web services. Then, they propose to automatically generate random source test cases from the WSDL specification and apply the metamorphic relations. A tool to partially automate the process is presented. Their approach is evaluated with three subject web services and mutation analysis.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Web services					
Application scenarios					Number of MRs	
Automatic Teller Machine					6	
Seismic query service					12	
Money numerical quantity to text					4	
				Total:	22	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Balance Transfer (ATM) service	Java	136	☐	200	129	0
Seismic web service	Java	551	☒	100	724	0
RMB converter service	NR	NR	☐	100	195	0
Total				400	1048	0
Source TCs generation technique:	Random					
Evaluation metrics:	Mutation Score (MS) and Fault Discovery Rate (FDR)					
☐ Available evaluation material						
Lessons learned / guidelines						
- Good MRs are relations which involve the execution of the core functionality. - Good MRs are those that can make the multiple executions of the program as different as possible.						
Challenges						

B.80 Xie et al. IST'12

See legend in page 24 to know the exact meaning of each field.

2012-xie-ist						
Publication data						
Authors:	X. Xie and W. E. Wong and T. Y. Chen and B. Xu					
Title:	Metamorphic slice: An application in spectrum-based fault localization					
Publication:	Information and Software Technology					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2012					
DOI/URL:	http://dx.doi.org/10.1016/j.infsof.2012.08.008					
Pages:	14					
Country:	Australia					
Contact:	xxie@swin.edu.au					
Summary:						
This article proposes the combination of MT and Spectrum-Based Fault Localization (SBFL) for program debugging. More specifically, the authors use a new concept, *metamorphic slice*, resulting from the integration of MT and program slicing. Instead of using conventional program slices and the failure or pass information from test cases, metamorphic slices allow working with violation or non-violation information from MRs in programs without oracles. A case study with 9 programs and mutation analysis is presented. Also, two real bugs are detected in two of the programs of the Siemens Suite. The results suggest that the approach is applicable offering a fault detection capability similar to the conventional SBFL techniques. This article is an extension of a previous conference paper (Xie et al. 2011 QSIC)						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:	Spectrum fault localization					
Application domain(s):	Pattern matching, lexical analyser, priority scheduler, altitude separation, information measure, bioinformatics.					

Application scenarios	Number of MRs
grep – pattern matching	3
Short sequence mapping (bioinformatics)	3
Lexical analyser	3
Priority scheduler	3
Altitude separation	3
Information measure	3
String matching	3
Total:	21

Evaluation

Program	Language	Size	Real	STCs	Mutants	Faults
print_tokens	C	342	☐	4130	77 max[1]	1
print_tokens2	C	355	☐	4115	79 max	
replace	C	512	☐	5542	144 max	
schedule	C	292	☐	2650	98 max	1
Schedule2	C	269	☐	2710	94 max	
tcas	C	135	☐	1608	69 max	
tot_info	C	273	☐	1052	76 max	
seqMap v1.0.8	C++	1783	☒	300	97 max	

[1] The exact number of mutants is not reported. This is the maximum number of mutants according to the number of mutants used for each MR on each program.

grep 1.2	C	7309	☒	10069	146 max	
Total				29466	880 max	2
Source TCs generation technique:	Test suite and random testing.					
Evaluation metrics:	EXAM score (percentage of executable statements that have to be examined until the first statement containing the bug is reached)					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.81 Yi et al. ACSIE'12

See legend in page 24 to know the exact meaning of each field.

2012-yi-acsie						
Publication data						
Authors:	Y.Yao and S. Huang and M. Ji					
Title:	Research on Metamorphic Testing for Oracle Problem of Integer Bugs					
Publication:	Fourth International Conference on Advances in Computer Science and Information Engineering					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2012					
DOI/URL:	http://dx.doi.org/10.1007/978-3-642-30126-1_16					
Pages:	6					
Country:	China					
Contact:	yaoyi226@yahoo.com.cn					
Summary: This paper proposes using MT for the detection of Integer bugs. A small case study with a traffic collision avoidance program, one MR and 15 mutants is presented. The results suggest that MT is more effective than the formal safety property method.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Security software (Integer bug detection)					
Application scenarios					Number of MRs	
Traffic collision avoidance system					1	
				Total:	1	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
TCAS	C	173	☐	60	15	0
			☐			
Total		173			15	0
Source TCs generation technique:	Not reported					
Evaluation metrics:	Failure detection ratio					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.82 Cao et al. QSIC'13

See legend in page 24 to know the exact meaning of each field.

2013-cao-qsic						
Publication data						
Authors:	Y. Cao and Z. Zhou and T. Y. Chen					
Title:	On the Correlation between the Effectiveness of Metamorphic Relations and Dissimilarities of Test Case Executions					
Publication:	13th International Conference on Quality Software					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2013					
DOI/URL:	http://dx.doi.org/10.1109/QSIC.2013.43					
Pages:	10					
Country:	Australia					
Contact:	zhiquan@uow.edu.au					
Summary: This paper assesses the correlation between fault-detection effectiveness of MRs and test case dissimilarity. An extensive experiment is reported using 83 faulty programs and 7 distance metrics in test cases. The results show that there is a strong and statistically significant correlation between the fault-detection capability of MRs and the distance among test cases, especially when using the Branch Coverage Manhattan Distance (BCMD) metric.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☐ Case study / application ☒ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Graph theory, pattern matching, text transformation, command line interpreter					

Application scenarios	Number of MRs
Dijkstra's shortest path algorithm	20
Critical path search in a directed graph	20
Shortest and second shortest path in a graph	20
Calculator for large integers	43
Pattern matching (grep)	10
Stream editor (performs text transformations in an input stream)	33
Command language interpreter (bash)	10
Total:	156

Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
spWiki	C	95	☐	1000	19	0
cpWiki	C	125	☐	1000	18	0
spStudent	C++	200	☐	1000	0	10
bigInt	C++	500	☐	1000	0	21
grep	C	1006	☒	10000	5	0
sed	C	1442	☒	4333	7	0
bash	C	5984	☒	10000	6	0
Total		8526		28333	55	31
Source TCs generation technique:	Test suite and random testing.					
Evaluation metrics:	Failure detection rate					
☐ Available evaluation material						

Lessons learned / guidelines
Challenges

B.83 Chan and Tse QSIC'13

See legend in page 24 to know the exact meaning of each field.

2013-chan-qsic						
Publication data						
Authors:	W. K. Chan and T. H. Tse					
Title:	Oracles Are Hardly Attain'd, and Hardly Understood: Confessions of Software Testing Researchers					
Publication:	The Symposium on Engineering Test Harnesses					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2013					
DOI/URL:	http://dx.doi.org/10.1109/QSIC.2013.16					
Pages:	8					
Country:	Hong Kong					
Contact:	wkchan@cityu.edu.hk					
Summary:						
This paper summarizes the authors' works on the oracle problem focusing on three scenarios: i) testing without a mechanism to determine the expected output, ii) testing without a mechanism to gauge the actual output, and iii) testing without a mechanism to decide whether the actual results agree with the expected outcomes. Several previously published works on MT are presented to illustrate their contributions to scenarios i) and ii).						
Contribution						
☐ New technique / method ☒ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Numerical programs, ubiquitous computing					
Application scenarios					Number of MRs	
Partial differential equations					1	
Smart delivery system					2	
				Total:	3	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.84 Dong et al. ICESS'13

See legend in page 24 to know the exact meaning of each field.

2013-dong-icsess						
Publication data						
Authors:	G. Dong and T. Guo and P. Zhang					
Title:	Security Assurance with Program Path Analysis and Metamorphic Testing					
Publication:	4th IEEE International Conference on Software Engineering and Service Science					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2013					
DOI/URL:	http://dx.doi.org/10.1109/ICSESS.2013.6615286					
Pages:	5					
Country:	China					
Contact:	donggw@itsec,gov.cn					
Summary:	This paper proposes using symbolic execution and program path analysis to design MRs that cover all the program paths with a few executions as possible. The feasibility of the approach is evaluated using two small case studies and mutation analysis.					
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:	Structural testing (path coverage), symbolic execution					
Application domain(s):	Numerical programs					
Application scenarios					Number of MRs	
Trisquare					2	
Normal distribution probability					3	
				Total:	5	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Trisquare	NR	NR	☐	NR	4	0
Normal distribution probability	NR	36	☐	NR	3	0
Total		36			7	0
Source TCs generation technique:	Random					
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.85 Hui et al. MPE'13

See legend in page 24 to know the exact meaning of each field.

2013-hui-mpe						
Publication data						
Authors:	Z. Hui and S. Huang and Z. Ren and Y. Yao					
Title:	Metamorphic Testing Integer Overflow Faults of Mission Critical Program: A Case Study					
Publication:	Mathematical Problems in Engineering					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2013					
DOI/URL:	http://dx.doi.org/10.1155/2013/381389					
Pages:	6					
Country:	China					
Contact:	hzw_1983821@163.com					
Summary:						
The paper proposes the use of metamorphic testing to detect faults related to integer overflows. A case study with the aircraft collision avoidance system TCAS from the Siemens Suite is presented. One metamorphic relations is proposed and evaluated using three mutants.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Aircraft conflict detection					
Application scenarios						Number of MRs
Aircraft collision avoidance system						1
				Total:		1
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
TCAS	C		☐	87	1	
			☐			
Total				87	1	
Source TCs generation technique:	Existing suite					
Evaluation metrics:	Fault detection ratio					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.86 Hui and Huang WCSE'13

See legend in page 24 to know the exact meaning of each field.

2013-hui-wcse						
Publication data						
Authors:	Z. Hui and S. Huang					
Title:	Achievements and Challenges of Metamorphic Testing					
Publication:	Fourth World Congress on Software Engineering					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2013					
DOI/URL:	http://dx.doi.org/10.1109/WCSE.2013.16					
Pages:	5					
Country:	China					
Contact:						
Summary: This paper informally reviews some of the previous works on MT in terms of i) construction of MRs, ii) selection of MRs, iii) generation of test cases, and iv) evaluation of MT effectiveness. As a result of their review, the authors identify several challenges on MT research.						
Contribution						
☐ New technique / method ☒ Survey / overview ☐ Empirical study ☐ Other: ☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):						
Application scenarios						Number of MRs
					Total:	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						
- Lack of formalized description method for MRs with consistency and ambiguity. - Lack of objective and efficient metrics for MRs. - Construct more efficient MRs based on basic ones to ensure the completeness of MRs.						

B.87 Hui and Huang WCSE'13 (b)

See legend in page 24 to know the exact meaning of each field.

2013-hui-wcse-b						
Publication data						
Authors:	Z. Hui and S. Huang					
Title:	A Formal Model for Metamorphic Relation Decomposition					
Publication:	Fourth World Congress on Software Engineering					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2013					
DOI/URL:	http://dx.doi.org/10.1109/WCSE.2013.14					
Pages:	5					
Country:	China					
Contact:						
Summary: This paper presents a formal model for the definition of MRs. In particular, they propose decomposing the definition of each relation in three parts: Input Relation (IR), Output Relation (OR) and program function or Self Relation (SR). Each part is defined using predicate logic. To illustrate their approach, the authors formalize some MRs found in the literature.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Numerical programs					
Application scenarios						Number of MRs
Sin						1
Integral						1
Shortest path						1
Determinant of a matrix						2
K-Nearest neighbors						2
Integer search in an ordered data structure						1
					Total:	8
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						
- MRs in different domains differ significantly and may not be easy to understand for software testers with different domain knowledge. Also, they can be ambiguous and hard to validate. Challenge: Define formal methods to describe MRs.						

B.88 Jiang et al. ICESS'13

See legend in page 24 to know the exact meaning of each field.

2013-jiang-icsess						
Publication data						
Authors:	M. Jiang and T. Y. Chen and F. Kuo and Z. Ding					
Title:	Testing Central Processing Unit scheduling algorithms using Metamorphic Testing					
Publication:	4th IEEE International Conference on Software Engineering and Service Science					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2013					
DOI/URL:	http://dx.doi.org/10.1109/ICSESS.2013.6615365					
Pages:	7					
Country:	China					
Contact:						
Summary: This paper proposes using MT for the detection of faults in CUP scheduling programs. Six MRs are presented for the Highest Response Ratio Next (HRRN) scheduler. An experimental evaluation with two simulators is reported. Two defects are detected in one of the simulators. Further experiments using mutation analysis suggests that the proposed MT approach is an effective method for testing HRRN schedulers.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Scheduling program					
Application scenarios						Number of MRs
Highest Response Ratio Next (HRRN) scheduling program						6
					Total:	6
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
HRRN Simulator 1	NR	NR	☐	500	10	0
HRRN Simulator 2	NR	NR	☒	500	10	2
Total				1000	20	2
Source TCs generation technique:	Random testing					
Evaluation metrics:	Effectiveness of MR (No of violated pairs of mutant and MR/ total number of used pairs) and MT (No. of violated metamorphic test groups/ total no. of executed metamorphic test groups)					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.89 Kanewala and Bieman ISSRE'13

See legend in page 24 to know the exact meaning of each field.

2013-kanewala-issre						
Publication data						
Authors:	U. Kanewala and J. M. Bieman					
Title:	Using Machine Learning Techniques to Detect Metamorphic Relations for Programs without Test Oracles					
Publication:	24th International Symposium on Software Reliability Engineering					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2013					
DOI/URL:	http://dx.doi.org/10.1109/ISSRE.2013.6698899					
Pages:	10					
Country:	United States					
Contact:	upuleegk@cs.colostate.edu					
Summary:						
This paper proposes using machine learning techniques for the automated generation of MRs for mathematical programs. The method works at the function level. First, a Control Flow Graph (CFG) is generated from the source code of the function. Then, a number of features are extracted from the CFG, and a machine learning algorithm uses these features to create a predictive model. An experimental evaluation is reported using 48 mathematical functions and three different types of MRs: permutative, additive and inclusive. Two different machine learning techniques are used: SVM and decision trees. Mutation analysis reveals that the generated MRs are effective in detecting 66% of the mutants.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Numerical programs					
Application scenarios						Number of MRs
Mathematical functions						NR
				Total:		
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
35 mathematical functions	Java	7-45	☐	350	988	0
			☐			
Total		7-45		350	988	0
Source TCs generation technique:	Random					
Evaluation metrics:	Mutation score					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.90 Kanewala and Bieman SECSE'13

See legend in page 24 to know the exact meaning of each field.

2013-kanewala-secse							
Publication data							
Authors:	U. Kanewala and J. M. Bieman						
Title:	Techniques for Testing Scientific Programs Without an Oracle						
Publication:	5th International Workshop on Software Engineering for Computational Science and Engineering						
Pub. Type:	☐ Journal ☐ Conference / Symp. ☒ Workshop ☐ Other:						
Year:	2013						
DOI/URL:	http://dx.doi.org/10.1109/SECSE.2013.6615099						
Pages:	10						
Country:	United States						
Contact:	upuleegk@cs.colostate.edu						
Summary: This paper examines three different testing techniques: MT, assertion checking and generation of oracles using machine learning. They compare the techniques in terms of i) oracle properties, ii) fault finding measures, iii) potential automation, and iv) required domain knowledge. For the comparison, authors review some works related to each technique discussing their main findings. For each technique, its limitations and unresolved problems are outlined. The paper concludes mentioning some of the tasks that could be potentially automated such as the automated generation of likely MRs and the elimination of spurious invariants.							
Contribution							
☐ New technique / method ☒ Survey / overview ☐ Empirical study ☐ Other: ☐ Case study / application ☒ Assessment ☐ Tool							
Combination with other techniques:	Assertion checking						
Application domain(s):	Numerical programs						
Application scenarios						Number of MRs	
Machine learning							
Sum of integers in an array						2	
JPEG encoder							
					Total:		
Evaluation							
Program	Language	Size	Real	STCs	Mutants	Faults	
			☐				
			☐				
Total							
Source TCs generation technique:							
Evaluation metrics:							
☐ Available evaluation material							
Lessons learned / guidelines							
- Not all MRs have the same fault detection ability. - MRs that enforce an equality relationship are preferred over MRs that enforces a non-equality relationship, since an equality relationship can be violated more easily than a non-equality relationship.							
Challenges							
- Automatically detecting likely MRs for a program. Minimize spurious relations. - Prioritization MRs. - Identify optimum combinations of MRs to reduce the number of executions required. - Identify limitation of MT. Are there faults that could never be detected using MT?							

B.91 Lei et al. QSIC'13

See legend in page 24 to know the exact meaning of each field.

2013-lei-qsic						
Publication data						
Authors:	Y. Lei and X. Mao and T. Y. Chen					
Title:	Backward-Slice-Based Statistical Fault Localization without Test Oracles					
Publication:	13th International Conference on Quality Software					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2013					
DOI/URL:	http://dx.doi.org/10.1109/QSIC.2013.45					
Pages:	10					
Country:	China					
Contact:	yanlei@nudt.edu.cn					
Summary:						
The paper proposes the integration of Backward-Slice-based Statistical Fault Localization (BSSFL) and MT to address the localization of bugs in programs with the oracle problem. BSSFL is an extension of Spectrum Fault Localization (SFL) in which backward slicing techniques are used to determine whether the executions of a statements affects (or do not affect) the outputs of test cases. The work is inspired in the work of Xie et al. (Xie et al 2012 IST) combining SFL and MT. The results of an extensive evaluation using 8 programs and mutation analysis is reported. The results suggest that the presented approach provides similar performance to that of conventional BSSFL techniques with available test oracles.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:	Backward-slice-based statistical fault localization (debugging)					
Application domain(s):	Lexical analyser, pattern recognition, pattern matching, information measurement, priority scheduler, altitude separation					
Application scenarios					Number of MRs	
Lexical analyser					3	
Pattern recognition					3	
Priority scheduler					3	
Altitude separation					3	
Information measure					3	
Pattern matching					3	
				Total:	18	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
print_tokens	C	342	☐	4130	355	0
print_tokens2	C	355	☐	4115	341	0
replace	C	512	☐	5542	279	0
schedule	C	292	☐	2650	275	0
schedule2	C	262	☐	2710	303	0
tcas	C	135	☐	1608	397	0
tot_info	C	274	☐	1052	94	0
grep v 2.0	C	7309	☒	10069	516	0
Total	C	9481		31876	2560	0
Source TCs generation technique:	Test suite + random testing					
Evaluation metrics:	EXAM					
☐ Available evaluation material						

Lessons learned / guidelines
Challenges

B.92 Rao et al. QSIC'13

See legend in page 24 to know the exact meaning of each field.

2013-rao-qsic						
Publication data						
Authors:	P. Rao and Z. Zheng and T. Y. Chen and N. Wang and K. Cai					
Title:	Impacts of Test Suite's Class Imbalance on Spectrum-Based Fault Localization Techniques					
Publication:	The Symposium on Engineering Test Harnesses					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2013					
DOI/URL:	http://dx.doi.org/10.1109/QSIC.2013.18					
Pages:	8					
Country:	China					
Contact:	peifengrao@163.com					
Summary:						
This paper presents an experimental evaluation of the impact of class imbalance (percentage of failure/pass test cases) in Spectrum-Based Fault Localization (SBFL) techniques using MT (Xie et al. 2012 IST). The evaluation is conducted on 8 programs using mutation analysis. Among other conclusions, the results suggest that the impact of class imbalance using metamorphic slices is similar for SBFL using conventional slices. As an additional result, a real defect is detected in one of the programs of the Siemens suite.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☒ Assessment ☐ Tool						
Combination with other techniques:	Spectrum-based fault localization					
Application domain(s):	Pattern matching, lexical analyser, priority scheduler, altitude separation, information measure.					
Application scenarios					Number of MRs	
grep – pattern matching					3	
Lexical analyser					3	
Priority scheduler					3	
Altitude separation					3	
Information measure					3	
String matching					3	
				Total:	18	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
print_tokens	C	472	☐	4130	46 max[1]	
print_tokens2	C	399	☐	4115	37 max	
replace	C	512	☐	5542	129 max	
schedule	C	292	☐	2650	64 max	
schedule2	C	301	☐	2710	48 max	1
tcas	C	440	☐	1608	24 max	
tot_info	C	141	☐	1052	46 max	
grep 1.2	C	15633	☒	1006	188 max	
Total		18190		3187	582 max	1
Source TCs generation technique:	Test suite + random					

[1] The exact number of mutants is not reported. This is the maximum number of mutants according to the number of mutants used for each MR on each program.

Evaluation metrics:	Risk evaluation formulas
☐ Available evaluation material	
Lessons learned / guidelines	
Challenges	

B.93 Yi et al. ISDEA'13

See legend in page 24 to know the exact meaning of each field.

2013-yi-isdea						
Publication data						
Authors:	Y. Yao and C. Zheng and S. Huang and Z. Ren					
Title:	Research on Metamorphic Testing: A Case Study in Integer Bugs Detection					
Publication:	Fourth International Conference on Intelligent Systems Design and Engineering Applications					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2013					
DOI/URL:	http://dx.doi.org/10.1109/ISDEA.2013.516					
Pages:	6					
Country:	China					
Contact:	yaoyi2266@163.com					
Summary:						
This paper proposes using MT for the detection of integer bugs. A small case study with a program for polygon area calculation is presented. A fault is manually seeded in the program and detected by 1 out of the 2 MRs proposed. Authors conclude that more research is needed to investigate the effectiveness of MT for the detection of integer bugs.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Numerical program					
Application scenarios					Number of MRs	
Calculate area and perimeter of a polygon					2	
				Total:	2	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Calculate triangle area (Heron's formula)	NR	NR	☐	10	1	0
			☐			
Total				10	1	0
Source TCs generation technique:	Random					
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.94 Aruna and Prasad ICACCI'14

See legend in page 24 to know the exact meaning of each field.

2014-aruna-icacci						
Publication data						
Authors:	C. Aruna and R. S. R. Prasad					
Title:	Metamorphic relations to improve the test accuracy of Multi Precision Arithmetic software applications					
Publication:	Second International Symposium on Women in Computing and Informatics					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2014					
DOI/URL:	http://dx.doi.org/10.1109/ICACCI.2014.6968586					
Pages:	5					
Country:	India					
Contact:						
Summary: This paper proposes using MT for the detection of precision faults in arithmetic software applications. Seven MRs for multiplication and division of multi arithmetic precision programs are presented. The work is evaluated with a small case study with five mathematical programs. Mutation analysis is mentioned although it is unclear how the mutants were generated and how many of them were derived from each program. The results are compared with "other system level approaches" although there are no references for them.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Numerical programs					
Application scenarios						Number of MRs
Multiplication on Multi Precision Arithmetic (MPA)						7
					Total:	7
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
SVM	C	NR	NR[1]	500	68	0
Arhant-II	C	NR	NR	500	19	0
GBT	C	NR	NR	500	24	0
PAYL	C	NR	NR	500	53	0
Total				2500	164	0
Source TCs generation technique:	Random					
Evaluation metrics:	Mutation score					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

[1] There are no references to the subject tools.

B.95 Aruna and Prasad ICT'14

See legend in page 24 to know the exact meaning of each field.

2014-aruna-ict						
Publication data						
Authors:	C. Aruna and R. S. R. Prasad					
Title:	Testing Approach for Dynamic Web Applications Based on Automated Test Strategies					
Publication:	48th Annual Convention of Computer Society of India- Vol II ICT and Critical Infrastructure					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2014					
DOI/URL:	http://dx.doi.org/10.1007/978-3-319-03095-1_43					
Pages:	12					
Country:	India					
Contact:	chittineni.aruna@gmail.com					
Summary:						
This paper proposes extending the Ochiai algorithm with MT for fault localization in dynamic web application. Five MRs for a classification algorithm are presented. It is unclear how this is related to the generation of effective test case with high fault-localization capability. Examples are missing. Some results graphs are presented although it is not clear how they were obtained, i.e. subject programs, experimental settings, etc.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:	Spectrum-based fault localization (Ochiai algorithm)					
Application domain(s):	Machine learning					
Application scenarios					Number of MRs	
Classification algorithm					5	
				Total:	5	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.96 Barr et al. TSE'14

See legend in page 24 to know the exact meaning of each field.

2014-barr-tse						
Publication data						
Authors:	E.T. Barr and M. Harman and P. McMinn and M. Shahbaz and S. Yoo					
Title:	The Oracle Problem in Software Testing: A Survey					
Publication:	IEEE Transactions on Software Engineering					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2014					
DOI/URL:	http://dx.doi.org/10.1109/TSE.2014.2372785					
Pages:	30					
Country:	United Kingdom					
Contact:	e.barr@ucl.ac.uk					
Summary: This article presents a survey on the oracle problem in software testing. Among other techniques, MT is briefly reviewed within the category of derived test oracles.						
Contribution						
☐ New technique / method ☒ Survey / overview ☐ Empirical study ☐ Other: ☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):						
Application scenarios						Number of MRs
				Total:		
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						
- Automated discovery of MRs.						

B.97 Carzaniga et al. ICSE'14

See legend in page 24 to know the exact meaning of each field.

2014-carzaniga-icse						
Publication data						
Authors:	A. Carzaniga and A. Goffi and A. Gorla and A. Mattavelli and M. Pezzè					
Title:	Cross-checking oracles from intrinsic software redundancy					
Publication:	International Conference on Software Engineering					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2014					
DOI/URL:	http://doi.acm.org/10.1145/2568225.2568287					
Pages:	12					
Country:	Switzerland					
Contact:	antonio.carzaniga@usi.ch					
Summary:						
The paper presents the concept of "cross-checking oracles". Given a source test case, the authors propose to generate a new test case in which one or more operations are replaced by redundant ones. If the output of both test cases are not equal, the code must contain a bug. The author propose an implementation of their approach using aspects. The identification of redundant methods is a manual task.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Object-oriented programs					
Application scenarios						Number of MRs
Unit testing (Metamorphic relation: equivalence)						1
					Total:	1
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Guava	Java	NR	☒	NR	1581	0
Joda-Time	Java	NR	☒	NR	842	0
GraphStream	Java	NR	☒	NR	998	1
Total					3421	1
Source TCs generation technique:	Test suite + random generation					
Evaluation metrics:	Mutation score					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.98 Goffi et al. FSE'14

See legend in page 24 to know the exact meaning of each field.

2014-goffi-fse						
Publication data						
Authors:	A. Goffi and A. Gorla and A. Mattavelli and M. Pezze and P. Tonella					
Title:	Search-based Synthesis of Equivalent Method Sequences					
Publication:	22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2014					
DOI/URL:	http://dx.doi.org/10.1145/2635868.2635888					
Pages:	11					
Country:	Switzerland					
Contact:	goffia@usi.ch					
Summary: This paper proposes a search-based algorithm for the automated generation of likely-equivalent method sequences in object oriented programs. The authors suggest that such likely-equivalent sequences could be used as MRs during testing. The approach was evaluated with 47 methods of 7 classes taken from the Stack Java Standard Library and the Graphstream library. The algorithm automatically synthesized 123 equivalent method sequences, which represent more than 87% of the 141 sequences that has been manually identified beforehand.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:	Search-based optimization					
Application domain(s):	Object-oriented programs					
Application scenarios					Number of MRs	
				Total:		
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.99 Goffi ICSEDS'14

See legend in page 24 to know the exact meaning of each field.

2014-goffi-icseds						
Publication data						
Authors:	A. Goffi					
Title:	Automatic generation of cost-effective test oracles					
Publication:	International Conference on Software Engineering (ICSE Doctoral symposium)					
Pub. Type:	☐ Journal ☐ Conference / Symp. ☐ Workshop ☒ Other:Doct. S.					
Year:	2014					
DOI/URL:	http://doi.acm.org/10.1145/2591062.2591078					
Pages:	4					
Country:	Switzerland					
Contact:	alberto.goffi@usi.ch					
Summary: This doctoral symposium paper summarizes the work of the author on the generation of oracles for object oriented programs. In particular, the author propose to identify equivalence sequences of methods, that is, code fragments that should produce identical output for any input. Then, given a unit test case, the author propose to create follow-up test cases by replacing one more statements with equivalent ones. If the output of source and follow-up test cases is not the same, a candidate fault has been detected. The contribution is not presented as a metamorphic testing approach but it can be considered as an intuitive application of the technique.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):						
Application scenarios					Number of MRs	
				Total:		
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.100 Kanewala ICSTDS'14

See legend in page 24 to know the exact meaning of each field.

2014-kanewala-icstds						
Publication data						
Authors:	U. Kanewala					
Title:	Techniques for Automatic Detection of Metamorphic Relations					
Publication:	IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops					
Pub. Type:	☐ Journal ☐ Conference / Symp. ☐ Workshop ☒ Other: PhD Symposium					
Year:	2014					
DOI/URL:	http://dx.doi.org/10.1109/ICSTW.2014.62					
Pages:	2					
Country:	United States					
Contact:	upuleegk@cs.colostate.edu					
Summary: This doctoral symposium paper describes the work of the author on the automated detection of likely MRs in scientific programs using machine learning techniques. The author describes his preliminary work published in ISSRE (Kanewala and Bieman 2013 ISSRE).						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Numerical programs					
Application scenarios						Number of MRs
					Total:	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.101 Le et al. PLDI'14

See legend in page 24 to know the exact meaning of each field.

2014-le-pldi						
Publication data						
Authors:	V. Le and M. Afshari and Z. Su					
Title:	Compiler validation via equivalence modulo inputs					
Publication:	Conference on Programming Language Design and Implementation					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2014					
DOI/URL:	http://doi.acm.org/10.1145/2666356.2594334					
Pages:	11					
Country:	United States					
Contact:	vmle@ucdavis.edu					
Summary:						
The paper presents an approach to test compilers by creating equivalent versions of the programs used as test inputs. Given a program and a set of input values, the authors propose to create equivalent versions of the program by profiling its execution and pruning unexecuted code. Once a program and its equivalent variant are generated, both are used as input of the compiler under test checking for inconsistencies in their results. The method has been used to detect 147 confirmed bugs in two real C compilers, GCC and LLVM. The authors do not explicitly mention metamorphic testing but their approach can be considered a specific application of the technique.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Compilers					
Application scenarios						Number of MRs
Code optimization						1
					Total:	1
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
GCC	C	NR	☒	NR	0	111
LLVM	C	NR	☒	NR	0	84
Total						195
Source TCs generation technique:	Test suite + random					
Evaluation metrics:	Number of real bugs detected.					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.102 Liu et al. ICSE'14

See legend in page 24 to know the exact meaning of each field.

2014-liu-icse						
Publication data						
Authors:	H. Liu and I. I. Yusuf and H. W. Schmidt and T. Y. Chen					
Title:	Metamorphic Fault Tolerance: An Automated and Systematic Methodology for Fault Tolerance in the Absence of Test Oracle					
Publication:	36th International Conference on Software Engineering Companion					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2014					
DOI/URL:	http://dx.doi.org/10.1145/2591062.2591109					
Pages:	4					
Country:	Australia					
Contact:	huai.liu@rmit.edu.au					
Summary: This paper introduces a new method called Metamorphic Fault Tolerance (MFT). In MFT, MRs are used to determine the trustworthiness of inputs in terms of the number of violations and non-violations of MRs. Also, if an output is judged as untrustworthy, MRs can be used to calculate the right output in certain scenarios.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:	Fault tolerance					
Application domain(s):						
Application scenarios					Number of MRs	
				Total:		
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.103 Liu et al. TSE'14

See legend in page 24 to know the exact meaning of each field.

2014-liu-tse						
Publication data						
Authors:	H. Liu and F. Kuo and D. Towey and T. Y. Chen					
Title:	How Effectively Does Metamorphic Testing Alleviate the Oracle Problem?					
Publication:	IEEE Transactions on Software Engineering					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2014					
DOI/URL:	http://dx.doi.org/10.1109/TSE.2013.46					
Pages:	19					
Country:	Australia					
Contact:	huai.liu@rmit.edu.au					
Summary:						
This article presents an empirical study to investigate the effectiveness of MT addressing the oracle problem. In particular, the authors intend to answer the following research questions: to what extent can MT alleviate the oracle problem; how easily and successfully can tester detect faults using MT; and where the key factors that influence the effectiveness of MT. For the study, several groups of undergraduate and postgraduate students from two different universities were recruited to identify MRs in 5 subject programs of algorithmic type. MT was compared to random testing with and without oracle. The study reveals that MT effectively alleviates the oracle problem. A number of lessons learned are reported.						
Contribution						
☐ New technique / method ☐ Survey / overview ☒ Empirical study ☐ Other: ☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Optimization, numerical programs, graph theory					

Application scenarios	Number of MRs
Finding the k nearest neighbours of a sample point	14
Minimizing a deterministic finite automaton	13
Solving the multiple knapsack problem	24
Multiplying two sparse matrices	22
Solving the set coverage problem using a greedy algorithm	15
Total:	88

Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
FindKNN	Java	153	☐	1000	698	0
MinimizeDFA	Java	929	☐	1000	1660	0
MultipleKnapsack	Java	808	☐	1000	1905	2
SparseMatrixMultiply	Java	259	☐	1000	212	1
SetCover	Java	211	☐	1000	258	0
Total		2360		5000	4773	3

Source TCs generation technique:	Random
Evaluation metrics:	- Oracle Imitation Measure (OIM) - Fault Detection Effectiveness (FDE) of each MR - Relation between FDE and number of MRs. - Relation between MRs identified by the same tester and number of killed mutants. - Relation between MRs identified by the same testing team and number of killed mutants. - Relation between FDE and number of testers.

☐ Available evaluation material
Lessons learned / guidelines
- The identification of a sufficient number of appropriate MRs for testing, even by inexperienced testers, was possible with a very small amount of training. - The cost-effectiveness of the approach could be enhanced through the use of more diverse MRs. - A small number (between 3 and 6) of diverse MRs, even those identified in an ad-hoc manner, had a similar fault-detection capability to a test oracle. - The diversity of MRs is more important than their quantity. - It is strongly recommended that a tester should take diversity into account when selecting MRs for testing.
Challenges

B.104 Nuñez and Hierons ATJ'14

See legend in page 24 to know the exact meaning of each field.

2014-nunez-at						
Publication data						
Authors:	A. Nuñez and R. M. Hierons					
Title:	A methodology for validating cloud models using metamorphic testing					
Publication:	Annals of telecommunications Journal					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2014					
DOI/URL:	http://dx.doi.org/10.1007/s12243-014-0442-7					
Pages:	9					
Country:	Spain					
Contact:	alberto.nunez@pdi.ucm.es					
Summary: This article presents an MT-based approach for validating cloud models. In particular, the authors propose using MRs to detect unexpected behaviour when simulating cloud provisioning and usage. A case study using two cloud models on the iCanCloud tool are presented. The authors propose three MRs to detect faults related to performance, functionality and energy awareness respectively. The results of the study suggest that the approach is effective in revealing poorly designed cloud system models.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Cloud computing (simulation)					
Application scenarios						Number of MRs
Cloud model						3
				Total:		3
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
2 cloud models on iCanCloud[1]			☐	100	0	0
			☐			
Total				100	0	0
Source TCs generation technique:	Heuristic algorithm					
Evaluation metrics:	Number of test cases that successfully fulfilled each MR					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

[1] The cloud models used as input for the simulation are the actual artefact under test.

B.105 Segura et al. STVR'14

See legend in page 24 to know the exact meaning of each field.

2014-segura-stvr						
Publication data						
Authors:	S. Segura and A. Durán and A. B. Sánchez and D. L. Berre and E. Lonca and A. Ruiz-Cortés					
Title:	Automated metamorphic testing of variability analysis tools					
Publication:	Software Testing, Verification and Reliability Journal					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2014					
DOI/URL:	http://dx.doi.org/10.1002/stvr.1566					
Pages:	26					
Country:	Spain					
Contact:	sergiosegura@us.es					
Summary:	This article presents a generic MT-based approach for the detection of faults in variability analysis tools. A novel method is proposed in which MRs are used to compute the actual output of follow-up test cases. This enables generating large variability models (inputs) and their corresponding set of configurations (potentially huge). The approach is evaluated by trying to detect faults in 15 real tools in the domains of feature models, CUDF document and SAT formulas. As a result, 19 real faults are detected.					
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☒ Tool						
Combination with other techniques:						
Application domain(s):	Software variability					
Application scenarios					Number of MRs	
Analysis of feature models					5	
Analysis of CUDF documents					4	
Boolean satisfiability					5	
				Total:	14	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
FaMa 1.1.2	Java	NR	☒	1000	0	4
FLAME	Prolog	NR	☐	1000	0	5
SPLAR	Java	NR	☒	1000	0	3
p2cudf 1.14	Java	NR	☒	1000	0	2
aspcudf 1.7	C++	NR	☒	1000	0	0
cudf-check 0.6.2-1		NR	☒	1000	0	0
Sat4j 2.3.1	Java	NR	☒	10000	0	0
Lingeling ala-b02		NR	☒	10000	0	0
Minisat 2.2		NR	☒	10000	0	0
Clasp 2.1.3		NR	☒	10000	0	0
Picosat 535		NR	☒	10000	0	0
Rsat 2.0		NR	☒	10000	0	0
March_ks 2007		NR	☒	10000	0	3
March_rw 2011		NR	☒	10000	0	1
Kcnfs 1.2		NR	☒	10000	0	1
Total				96000	0	19

Source TCs generation technique:	Random
Evaluation metrics:	Number of real faults detected.
☒ Available evaluation material	
Lessons learned / guidelines	
Challenges	

B.106 Sun et al. FCS'14

See legend in page 24 to know the exact meaning of each field.

2014-sun-fcs						
Publication data						
Authors:	C. Sun and Z. Wang and G. Wang					
Title:	A property-based testing framework for encryption programs					
Publication:	Frontiers of Computer Science Journal					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2014					
DOI/URL:	http://dx.doi.org/10.1007/s11704-014-3040-y					
Pages:	12					
Country:	China					
Contact:	casun@ustb.edu.cn					
Summary:						
This paper presents a case study on the use of MT for the detection of fault in encryption algorithms. Three MRs for two encryption algorithms (Hill and RSA) are presented and evaluated using mutation analysis. The authors conclude that the approach is effective in detecting faults.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Encryption algorithms					
Application scenarios						Number of MRs
Hill algorithm						3
RSA algorithm						1
					Total:	4
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Hill cipher program	C	74	☐	200	353	0
RSA program	C	28	☐	200	301	0
Total		102		400	654	0
Source TCs generation technique:	Random					
Evaluation metrics:	Mutation Score (MS), Fault Discovery Rate (FDR)					
☐ Available evaluation material						
Lessons learned / guidelines						
- The increase in the size of the test suites does not improve the fault detection capability when source test cases are randomly generated.						
Challenges						

B.107 Xie et al. QSIC'14

See legend in page 24 to know the exact meaning of each field.

2014-xie-qsic						
Publication data						
Authors:	X. Xie and J. Tu and T. Y. Chen and B. Xu					
Title:	Bottom-up Integration Testing with the Technique of Metamorphic Testing					
Publication:	14th International Conference on Quality Software					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2014					
DOI/URL:	http://dx.doi.org/10.1109/QSIC.2014.29					
Pages:	6					
Country:	Australia					
Contact:	xxie@swin.edu.au					
Summary: This paper proposes an integration MT method, which combines bottom-up integration testing and MT. Roughly speaking, the authors propose defining MRs based on the properties from different sub-components of the system to achieve better effectiveness and fault isolation. Testing is still conducted in the whole system as so there is no need for decomposing the systems and using stubs. A case study using mutation analysis on a filter Feature Selection (FS) algorithm integrated in the tool Weka is presented. The results support the benefits of the approach.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Machine learning					
Application scenarios						Number of MRs
Filter Feature Selection (FS) algorithm						10
				Total:		10
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Weka 3.6.10	Java	NR	☒	400	50	0
			☐			
Total				400	50	0
Source TCs generation technique:	Random					
Evaluation metrics:	Mutation score and percentage of mutants killed by each MR.					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.108 Zhang et al. ASE'14

See legend in page 24 to know the exact meaning of each field.

2014-zhang-ase						
Publication data						
Authors:	J. Zhang and J. Chen and D. Hao and Y. Xiong and B. Xie and L. Zhang and H. Mei					
Title:	Search-based Inference of Polynomial Metamorphic Relations					
Publication:	29th ACM/IEEE International Conference on Automated Software Engineering					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2014					
DOI/URL:	http://dx.doi.org/10.1145/2642937.2642994					
Pages:	12					
Country:	China					
Contact:	zhangjie12@sei.pku.edu.cn					
Summary:						
This paper proposes an approach to automatically inferring polynomial metamorphic relations by analysing multiple executions of a program under test (i.e. black-box approach). The problem is model as an optimization program and solved using a Particle Swarm Optimization (PSO) algorithm. Filtering is required to discard low-quality MRs. Filtering applies a large number of randomly generated test inputs to a program P, and records whether a likely MR is violated by each test input. If the MR is violated in a high percentage of cases, it is deemed. The work is evaluated inferring likely MRs for 189 functions from 4 commercial and open source mathematical libraries. The results show that the generated MRs are effective in detecting seeded faults.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☐ Case study / application ☐ Assessment ☒ Tool						
Combination with other techniques:	Search-based algorithm (Particular swarm optimization)					
Application domain(s):	Numerical programs					
Application scenarios					Number of MRs	
				Total:	100	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
For the generation of likely MRs:						
Apache Commons Mathematics Library 2.2	Java	1626	☒			
JDK 1.6	Java	NR	☒			
GSL 1.8	C/C++	7309	☒			
MATLAB R2012b	NR	NR	☒			
For the evaluation of the generated MRs (all functions belong to Apache Commons Math. Library 3.2):						
sin	Java	NR	☒	1000	17	0
cos	Java	NR	☒	1000	19	0
tan	Java	NR	☒	1000	18	0
log10	Java	NR	☒	1000	58	0
log1p	Java	NR	☒	1000	115	0
asinh	Java	NR	☒	1000	297	0
atan	Java	NR	☒	1000	94	0
abs_d	Java	NR	☒	1000	7	0
abs_f	Java	NR	☒	1000	7	0
abs_i	Java	NR	☒	1000	15	0

abs_l	Java	NR	☒	1000	15	0
Total				11000	662	0
Source TCs generation technique:	Random					
Evaluation metrics:	Mutation score					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.109 Aruna and Prasad ICACCE'15

See legend in page 24 to know the exact meaning of each field.

2015-aruna-icacce						
Publication data						
Authors:	C. Aruna and R.S.R. Prasad					
Title:	Adopting Metamorphic Relations to verify Non-Testable Graph Theory Algorithms					
Publication:	2nd International Conference on Advances in Computing and Communication Engineering (ICACCE)					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2015					
DOI/URL:	http://dx.doi.org/10.1109/ICACCE.2015.138					
Pages:	6					
Country:	India					
Contact:						
Summary: This paper presents a case study on the application of MT to two graph theory algorithms: shortest path and minimal spanning tree. Seven MRs are proposed and a small experiment is reported. The results of MT are compared to those of alternative testing tools (OSPF and ArcGIS) in terms of execution time and number of test cases generated.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Graph theory					
Application scenarios						Number of MRs
Shortest path						2
Minimal spanning tree						5
					Total:	7
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Dijkstra' algorithm shortest path	NR	NR	☐	NR		
Dijkstra' algorithm minimal spanning tree	NR	NR	☐	NR		
Total						
Source TCs generation technique:	NR					
Evaluation metrics:	Execution time, number of test cases generated.					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.110 Cañizares et al. ICCS'15

See legend in page 24 to know the exact meaning of each field.

2015-canizares-iccs						
Publication data						
Authors:	P. C. Cañizares and A. Nuñez and M. Nuñez and J.J. Pardo					
Title:	A Methodology for Designing Energy-Aware Systems for Computational Science					
Publication:	ICCS 2015 International Conference On Computational Science					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2015					
DOI/URL:	http://dx.doi.org/10.1016/j.procs.2015.05.438					
Pages:	5					
Country:	Spain					
Contact:	alberto.nunez@pdi.ucm.es					
Summary: This short paper presents same preliminary ideas on the use of MT for the detection of bugs related to energy consumption in cloud environments.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Cloud computing, computational science					
Application scenarios						Number of MRs
				Total:		
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.111 Chen AST'15

See legend in page 24 to know the exact meaning of each field.

2015-chen-ast						
Publication data						
Authors:	T.Y. Chen					
Title:	Metamorphic Testing: A Simple Method for Alleviating the Test Oracle Problem					
Publication:	10th International Workshop on Automation of Software Test					
Pub. Type:	☐ Journal ☐ Conference / Symp. ☒ Workshop ☐ Other:					
Year:	2015					
DOI/URL:	http://dx.doi.org/10.1109/AST.2015.18					
Pages:	2					
Country:	Australia					
Contact:	tychen@swin.edu.au					
Summary: Keynote summary. Brief overview of the technique, its applications and integration with other testing techniques.						
Contribution						
☐ New technique / method ☒ Survey / overview ☐ Empirical study ☐ Other: ☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):						
Application scenarios						Number of MRs
					Total:	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.112 Chen et al. JSS'15

See legend in page 24 to know the exact meaning of each field.

2015-chen-jss						
Publication data						
Authors:	T. Y. Chen and P. Poon and X. Xie					
Title:	METRIC: METamorphic Relation Identification based on the Category-choice framework					
Publication:	The Journal of Systems and Software					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2015					
DOI/URL:	http://dx.doi.org/10.1016/j.jss.2015.07.037					
Pages:	14					
Country:	Australia					
Contact:	drpoonpl@yahoo.com.hk					
Summary:						
This article presents a specification-based methodology and associated tool called METRIC for the identification of MRs based the category-choice framework. The approach requires processing the program specification to partition the input domain in terms of categories, choices and complete test frames. Categories are mainly related to input parameters, choices to parameter values and test frames to valid combination of choices. The complete set of test frames is therefore an abstract representation of all the potential test cases of the system under test. Given a set of input test frames, METRIC guides testers on the identification of MR and related source and follow-up test cases. The results of an empirical study with 19 participants suggest that METRIC is effective and efficient at identifying MRs.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:	Category-choice framework					
Application domain(s):	Information systems					
Application scenarios						**Number of MRs**
Parking fee system						3
Company car and expense claim system (CAR)						~40
Meal ordering system (MOS)						~40
					Total:	
Evaluation						
Program	**Language**	**Size**	**Real**	**STCs**	**Mutants**	**Faults**
SCAR (construction of MRs only)			☒			
SMOS (construction of MRs only)			☒			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
- MRs that simultaneously consider both inputs and outputs of the SUT are harder to identify than those that focus on input relations exclusively.						
Challenges						

B.113 Hui et al. STA'15

See legend in page 24 to know the exact meaning of each field.

2015-hui-compsac						
Publication data						
Authors:	Z. Hui and S. Huang and H. Li and J. Liu and L. Rao					
Title:	Measurable Metrics for Qualitative Guidelines of Metamorphic Relation					
Publication:	7th IEEE International Workshop on Software Test Automation (STA)					
Pub. Type:	☐ Journal ☐ Conference / Symp. ☒ Workshop ☐ Other:					
Year:	2015					
DOI/URL:	http://dx.doi.org/10.1109/COMPSAC.2015.179					
Pages:	6					
Country:	China					
Contact:						
Summary:						
This paper presents three metrics to quantitatively assess the quality of MRs for numerical programs, namely: i) Number of inputs (InD(IR)), number of output relations (AC(OR)) and distance between inputs (Dis(IR)). Experimental results with four small programs with seeded faults are reported. The results seem inconclusive.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Numerical programs					
Application scenarios					Number of MRs	
Sine					17	
Area of a triangle					16	
Aircraft conflict detection					37	
Grade computation					9	
				Total:	79	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Sin	C/C++	99	☐		100	
Trisquare	C/C++	168	☐		100	
TCAS	C/C++	206	☐		100	
Grade	C/C++	2035	☐		300	
Total		2508			600	
Source TCs generation technique:	Random testing					
Evaluation metrics:	Fault detection ratio					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						
- Prioritization/assessment of MRs.						

B.114 Jameel et al. SNPD'15

See legend in page 24 to know the exact meaning of each field.

2015-jameel-snpd						
Publication data						
Authors:	T. Jameel and M. Lin and L. Chao					
Title:	Test Oracles Based on Metamorphic Relations for Image Processing Applications					
Publication:	16th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2015					
DOI/URL:	http://dx.doi.org/10.1109/SNPD.2015.7176238					
Pages:	6					
Country:	China					
Contact:	tahir@nlsde.buaa.edu.cn					
Summary: This paper presents a case study on the application of metamorphic testing to detect faults in morphological image operation such as dilation and erosion. Eight MRs are reported and assessed on the detection of seeded faults in a MATLAB erosion function.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other: ☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Image processing					
Application scenarios					Number of MRs	
Dilation and erosion					8	
				Total:	8	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
ImageDilation	C/C++		☒		33	
			☐			
Total					33	
Source TCs generation technique:	Random testing					
Evaluation metrics:	Number (and ratio) of mutants detected by each MR					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.115 Jin et al. COMPSAC'15

See legend in page 24 to know the exact meaning of each field.

2015-jin-compsac				
Publication data				
Authors:	H. Jin and Y. Jiang and N. Liu and C. Xu and X. Ma and J. Lu			
Title:	Concolic Metamorphic Debugging			
Publication:	COMPSAC Symposium on Software Engineering Technologies and Applications			
Pub. Type:	☐ Journal	☒ Conference / Symp.	☐ Workshop	☐ Other:
Year:	2015			
DOI/URL:	http://dx.doi.org/10.1109/COMPSAC.2015.79			
Pages:	10			
Country:	China			
Contact:	changxu@nju.edu.cn			

Summary:

This paper presents an approach called Concolic Metamorphic Debugging (Comedy) that integrate concolic testing, metamorphic testing and branch witching debugging to localize potential bugs. Comedy explores all possible programs paths in depth-first-order searching for the first one that pass the metamorphic relation which is used an oracle. The final goal is to isolate a minimum amount of code to obtain a passing input and use that isolation point to localize the fault. The approach, implemented in a tool, is evaluated with 21 algorithmic programs using mutation testing.

Contribution

☒ New technique / method	☐ Survey / overview	☐ Empirical study	☐ Other:
☐ Case study / application	☐ Assessment	☒ Tool	

Combination with other techniques:	Concolic testing, branch-switching debugging
Application domain(s):	Numerical programs, graph theory, strings.

Application scenarios	Number of MRs
Remove redundant whitespaces in an URI - Tomcat example	1
Finding a pair of points with the smallest Euclidean distance between them.	1
Shortest path	2
Maximum flow algorithm	2
Maximum rectangle	1
String search (BoyerMoore)	1
SurroundedRegion in an 2D board	1
DecodingWays (Given an encoded message containing digits, determine the total number of ways to decode it)	1
Largest rectangle	1
Max tree path sum	1
Enhanced edit distance	1
Interleaving string	1
Heap sort	1
Search in Rotated Sorted Array	2
Quick sort	1
First missing positive	2
Find Minimum in Rotated Sorted Array	2
2D matrix search	1
Distinct subsequence	1
Multi-segment MAXSUM	1
Kadane's MAXSUM	2
Prim's algorithm	3
Total:	30

Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
ClosestPair	Java	370	☐		281	
Dijkstra	Java	271	☐		214	
Edmonds-Karp	Java	229	☐		46	
MaximumRectangle	Java	113	☐		172	
BoyerMoore	Java	93	☐		71	
SurroundedRegion	Java	78	☐		309	
DecodingWays	Java	78	☐		409	
LargestRectangle	Java	77	☐		139	
MaxTreePathSum	Java	74	☐		76	
EditingDistance	Java	73	☐		223	
InterleavingString	Java	73	☐		182	
HeapSort	Java	66	☐		112	
Multi-MAXSUM	Java	61	☐		166	
SearchInRot	Java	53	☐		208	
QuickSort	Java	49	☐		75	
FirstMissingPositive	Java	40	☐		83	
MinInRot	Java	34	☐		71	
2D-MatrixSearch	Java	34	☐		38	
DistinctSubsequence	Java	32	☐		53	
MAXSUB	Java	25	☐		46	
Prim	Java	765	☐		620	
Total		2688			3594	
Source TCs generation technique:	Random testing					
Evaluation metrics:	Branch distance					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.116 Kanewala et al. STVR'15

See legend in page 24 to know the exact meaning of each field.

2015-kanewala-stvr						
Publication data						
Authors:	U. Kanewala and J. M. Bieman and A. Ben-Hur					
Title:	Predicting metamorphic relations for testing scientific software: a machine learning approach using graph kernels					
Publication:	Journal of Software Testing, Verification and Reliability					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2015					
DOI/URL:	https://dx.doi.org/10.1002/stvr.1594					
Pages:	25					
Country:	United States					
Contact:	upuleegk@cs.montana.edu					
Summary:						
This article presents a machine learning approach to predict metamorphic relations in numerical programs. This is an extension of a previous work of the authors (Kanewala and Bieman, 2013 ISSRE). The main novelty is the use of graph kernels to represent control flow and data dependency information. These graphs provide various way of measuring similarity and this was exploited to predict MRs under the intuition that programs with similar control flow and data graphs may have similar MRs. Their approach was evaluated trying to identify six different types of MRs in a corpus of 100 numerical programs.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Numerical programs					
Application scenarios						Number of MRs
Mathematical functions (100)						
				Total:		
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
The Colt project (for prediction)	Java		☒			
Apache Mahout (for prediction)	Java		☒			
Apache commons (for prediction)	Java		☒			
Total						
Source TCs generation technique:						
Evaluation metrics:	Balanced success rate, area under the curve					
☒ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.117 Lindvall et al. ICSE'15

See legend in page 24 to know the exact meaning of each field.

2015-lindvall-icse						
Publication data						
Authors:	M. Lindvall and D. Ganesan and R. Ardal and R. E. Wiegand					
Title:	Metamorphic Model-based Testing Applied on NASA DAT –an experience report					
Publication:	IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE)					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2015					
DOI/URL:	http://dx.doi.org/10.1109/ICSE.2015.348					
Pages:	10					
Country:	United States					
Contact:	mikli@fc-md.umd.edu					
Summary:						
The paper presents an experience report on the use of metamorphic testing to address acceptance testing of NASA's Data Access Toolkit (DAT). DAT is a huge database of telemetry data collected from different NASA missions, and an advance query interface to search and mine the available data. Due to the massive amount of data contained in the database, checking the correctness of the query results is challenging due to the oracle problem. To this purpose, metamorphic testing is used by formulating the same query in different equivalent ways and asserting that the resulting datasets are the same. A number of real issues detected with this approach are reported.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:	Model-based testing					
Application domain(s):	Database query					
Application scenarios					Number of MRs	
Database query processing					10	
				Total:	10	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
NASA Data Access Toolkit			☒			7
			☐			
Total						7
Source TCs generation technique:	Random testing					
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.118 Su et al. AST'15

See legend in page 24 to know the exact meaning of each field.

2015-su-ast						
Publication data						
Authors:	F. Su and J. Bell and C. Murphy and G. Kaiser					
Title:	Dynamic Inference of Likely Metamorphic Properties to Support Differential Testing					
Publication:	10th International Workshop on Automation of Software Test (AST)					
Pub. Type:	☐ Journal ☐ Conference / Symp. ☒ Workshop ☐ Other:					
Year:	2015					
DOI/URL:	http://dx.doi.org/10.1109/AST.2015.19					
Pages:	5					
Country:	United States					
Contact:	mikefhsu@cs.columbia.edu					
Summary:						
The paper proposes a novel approach, KABU, for the dynamic inference of likely metamorphic relations. The approach is inspired by previous work on inferring likely program invariants with programs as Daikon. The inference process was constrained by searching for a set of predefined metamorphic relations. A Java tool implementing the approach was presented and evaluated on the inference of likely metamorphic relations in two sample programs. As a result, KABU found more likely metamorphic relations than a group of 23 students trained in the task. Authors also proposed a method, Metamorphic Differential Testing (MDT), built upon KABU, to compare the metamorphic relations between different versions of the same program reporting the differences as potential bugs. The preliminary results on different versions of two classification algorithms detected the changes reported in the logs of the Weka library.						
Contribution						
☒ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☒ Tool						
Combination with other techniques:						
Application domain(s):	Numerical program, strings					
Application scenarios						Number of MRs
Knapsack						31
Superstring						16
					Total:	47
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Knapsack	Java		☐			
Superstring	Java		☐			
LogitBoost (Weka)	Java		☒			
Decorate (Weka)	Java		☒			
Total						
Source TCs generation technique:	Existing suite ("iris dataset provided by Weka")					
Evaluation metrics:	Identification rate					
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						

B.119 Zhou et al. TSE'15

See legend in page 24 to know the exact meaning of each field.

2015-zhou-tse						
Publication data						
Authors:	Z. Zhou and S. Xiang and T. Y. Chen					
Title:	Metamorphic Testing for Software Quality Assessment: A Study of Search Engines					
Publication:	IEEE Transactions on Software Engineering					
Pub. Type:	☒ Journal ☐ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2015					
DOI/URL:	http://dx.doi.org/10.1109/TSE.2015.2478001					
Pages:	22					
Country:	Australia					
Contact:	zhiquan@uow.edu.au					
Summary:						
This article presents a user-oriented metamorphic testing approach to test online search engines with two novel ideas. First, MRs are defined from the user perspective, representing the properties that they expect form the search engines, regardless of how the engine is designed. In practice, this means that MRs cannot only be used to detect faults in the software under test (verification) but also to check whether the program behaves as the user expect (validation). Second, it is argued that MRs can be used to evaluate quality related properties such as reliability, usability or performance. Five MRs are presented and used to automatically test 4 search engines revealing a number of failures.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☒ Case study / application ☐ Assessment ☐ Tool						
Combination with other techniques:						
Application domain(s):	Web search engines					
Application scenarios						Number of MRs
Web search						5
				Total:		5
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
Google	Online		☒			3
Bing	Online		☒			2
Chinese Bing	Online		☒			1
Baidu	Online		☒			1
Total						7
Source TCs generation technique:	Random generation (using dictionaries)					
Evaluation metrics:	ROCOF (rate of occurrence of failure) ROCOA (rate of occurrence of anomaly)					
☐ Available evaluation material						
Lessons learned / guidelines						
- In order to create an exhaustive list of MRs, follow-up test cases should not only be related to the source test input but also to the source test output.						
Challenges						

B.120 Zhu TSA'15

See legend in page 24 to know the exact meaning of each field.

2015-zhu-tsa						
Publication data						
Authors:	H. Zhu					
Title:	JFuzz: A Tool for Automated Java Unit Testing Based on Data Mutation and Metamorphic Testing Methods					
Publication:	Second International Conference on Trustworthy Systems and Their Applications					
Pub. Type:	☐ Journal ☒ Conference / Symp. ☐ Workshop ☐ Other:					
Year:	2015					
DOI/URL:	http://dx.doi.org/10.1109/TSA.2015.13					
Pages:	8					
Country:	United Kingdom					
Contact:	hzhu@brookes.ac.uk					
Summary:						
This paper presents JFuzz, a Java unit testing tool using metamorphic testing. In JFuzz, tests are specified in three parts, namely i) source test case inputs, ii) possible transformations on the test inputs, and iii) metamorphic relations as code assertions. Once these elements are defined, the tool automatically generates follow-up test cases by applying the transformations to the source test inputs, it executes source and follow-up test cases, and checks whether the metamorphic relations are violated.						
Contribution						
☐ New technique / method ☐ Survey / overview ☐ Empirical study ☐ Other:						
☐ Case study / application ☐ Assessment ☒ Tool						
Combination with other techniques:						
Application domain(s):	Numerical programs					
Application scenarios					Number of MRs	
Triangle classification program					1	
Sine					1	
				Total:	2	
Evaluation						
Program	Language	Size	Real	STCs	Mutants	Faults
			☐			
			☐			
Total						
Source TCs generation technique:						
Evaluation metrics:						
☐ Available evaluation material						
Lessons learned / guidelines						
Challenges						