Received: 14 January 2022 | Revised: 22 August 2022 | Accepted: 29 August 2022

DOI: 10.1002/smr.2509

REVIEW ARTICLE

Journal of Software: Evolution and Process WILEY

Metamorphic relation automation: Rationale, challenges, and solution directions

Emran Altamimi | Abdullah Elkawakjy | Cagatay Catal

Department of Computer Science and Engineering, Qatar University, Doha, Qatar

Correspondence
Cagatay Catal, Department of Computer Science and Engineering, Qatar University, Doha, Qatar.
Email: ccatal@qu.edu.qa

Funding information
Qatar National Research Fund, Grant/Award Number: NPRP12C-33905-SP-66

Abstract

Metamorphic testing addresses the issue of the oracle problem by comparing results transformation from multiple test executions. The relationship that governs the output transformation is called metamorphic relation. Metamorphic relations require expert knowledge and the generation of them is considered a time-consuming task. Researchers have proposed various techniques to automate metamorphic testing, generation, and selection. Although there are several research articles on this issue, there is a lack of overview of the state-of-the-art of metamorphic relation automation. As such, we performed a systematic literature review study to collect, extract, and synthesize the required data. Based on our research questions, the literature was categorized and summarized into different categories. We found that the automation of metamorphic relation is most effective in mathematical and scientific applications. We concluded that some approaches involve analysis of different forms of software-related information such as control flow graph and program dependence graph as well as an initial set of metamorphic relations. On the other hand, other methods involve analysis of executions of the software functions with random and specific inputs. The results show that this field is still in its infancy with opportunities for novel work, especially in methods utilizing machine learning.

KEYWORDS
automation, machine learning, metamorphic relations, metamorphic testing

1 | INTRODUCTION

Nowadays, software is pervasive and faults in software cause massive catastrophes,[1] and to avoid malfunctions and unstable software, software testing became vital in the software development process. The software testing process mostly involves the construction of test cases, which are then assessed based on outcomes.[2]

The test cases are generally assessed based on an oracle. Generally, the idea of test oracles is to make a judgment about the correctness of software. Broadly speaking, there are three types of test oracles: test verdicts, expected output, and metamorphic relations. However, the oracle might not exist or is very hard to construct in many software applications.[3] This problem is referred to as the oracle problem. When the oracle problem occurs, the usability and utility of several test case selection processes become severely limited, as determining whether a fault has been recognized or not is incredibly difficult, if not impossible.

Automated testing necessitates the use of automated test oracles, which may or may not exist. When an oracle is unavailable or too complicated to develop, the software is regarded as non-testable. This problem is most common in scientific applications, which are regarded as non-testable programs.[4] In such circumstances, a consultant should validate that the output of the program is appropriate for a given set of inputs. Sanders et al.[5] have noted that scientists do testing in an ad hoc manner. As a result, small problems, such as one-time failures, are more difficult to identify, and automation in testing is limited. Metamorphic testing[4] solves this basic problem by examining whether the software functions according to relationships between both the input and output rather than a predetermined test oracle.

Metamorphic relationships are relationships between numerous inputs and outputs. Whenever there is a discrepancy between how the output changes and what the anticipated metamorphic relation implies, it is termed a violation of the metamorphic relation. However, under the assumption that the metamorphic relation is correct, the fact that a metamorphic relation has been satisfied does not ensure that the software will operate correctly. In the opposite case, however, a break in a metamorphic relation indicates that the software has faults. In prior studies, it has been found that metamorphic testing can be a beneficial strategy for evaluating systems that do not include oracles.[3,6]

The basic process of metamorphic testing can be summarized as follows:

1. determine an appropriate metamorphic relation set to check for your program testing;
2. use testing approaches and methods such as random, structural, and fault-based methods to create a collection of initial test cases;
3. include a corresponding set of follow-up tests in any initial test case. This is done by performing the input modifications needed by the established metamorphic relations in Step 1; and
4. determine if the metamorphic relation's assertion is correct, run the first and follow-up test case pairs. When a runtime failure of a metamorphic relation is discovered, it indicates that the program being tested has a bug or a number of problems.

Metamorphic testing has been used in many areas such as bioinformatics[7] and search engines.[8] A common tool in bioinformatics is automated protein function prediction, which suffers from the oracle problem. The authors in Shahri et al.[7] tackled this issue by defining multiple domain-specific metamorphic relations. The metamorphic relations specify that there should be a change in the predicted gene ontology terms between the canonical protein sequence and their variants. Source and follow-up test cases were created based on the defined metamorphic relations. The authors in Segura et al.[37] reviewed 119 articles and found that metamorphic testing has been most extensively utilized in web services and applications, computer graphics, and simulation and modeling, making up 40% of all publications. In web services and applications, for example, metamorphic testing was utilized for both verification (detection of software's faults) and validation (validating the software compared to the user expectations).[8] The authors[8] achieved this by incorporating user expectations of the test results in the process, where metamorphic relations are identified and defined from a functional perspective. Two categories of metamorphic relations were identified: no missing pages and ranking consistency. Other quality properties can also be tested such as performance, usability, and reliability. In computer graphics, metamorphic testing was utilized to compare random image generation methods that are commonly used for testing image processing programs.[38] Another case study utilized metamorphic testing for fault detection in applications of morphological image operations (for example, dilation and erosion). Some use cases of metamorphic testing in simulation and modeling include the detection of faults in casting simulations,[39] the conformance between network protocols and network simulators,[40] and the detection of unexpected behavior when simulating cloud provisioning and usage.[41] Metamorphic testing, however, suffers from two main limitations: identifying metamorphic relations and selecting which ones to test. These two concerns make metamorphic testing a highly difficult operation and demand extensive knowledge of the software.

Metamorphic testing is easy to automate if the set of metamorphic relations is available. In the automatic generation of test cases, the construction is done using existing methods with follow-up cases constructed via the metamorphic relations. In execution and verification, the automation is straightforward with simple scripts to run the tests and compare results. This makes metamorphic testing highly lucrative as it lowers the cost of testing while solving the oracle problem faced with so many application domains. However, a significant hurdle to fully automate metamorphic testing is the identification and optimum selection of metamorphic relations, which incur a marginal cost.

To tackle these challenges, numerous techniques were developed to automate these operations. However, it is currently unclear to what extent the current literature aids in solving these issues. It is also not evident what are the prerequisites of such techniques, for example, do they require a labeled dataset? Do they require human input or expert knowledge (i.e., what is the degree of automation)? Do they require white or black-box access to the software?

Using a systematic literature review (SLR) methodology, we have searched electronic databases to find relevant papers and we extracted the data that can help answer our research questions. To the best of our knowledge, there is no SLR paper on automating metamorphic relations in the literature and therefore, this study fills that gap for future research. We followed the SLR protocol proposed in Kitchenham et al.[9] and the guidelines discussed in Tummers et al.,[10] and we were able to retrieve 429 papers, of which 19 articles met the selection criteria and were judged to be of high quality. Research in this area will benefit from this publication, which lays the groundwork for future studies on the automated detection, creation, and selection of metamorphic relations. The following sections are explained as follows: Section 2 provides context and related work. Section 3 describes the chosen research strategy in depth. Section 4 presents the SLR findings. The discussion is in Section 5. Conclusions and future research are discussed in Section 6.

2 | BACKGROUND AND RELATED WORK

Metamorphic testing initially described by Chen et al.[11] can test programs when the oracle does not exist or demands excessive resources to construct. Metamorphic testing executes the program numerous times and compares the outcomes rather than looking at individual results. The comparison and assessment are based on metamorphic relations, which are key aspects of the program. For example, in a search engine, when given two search queries, a general one and a more specific search, the latter search query results should be a subset of the former search query results, otherwise, it is considered that the software has failed. As a result, metamorphic relations are regarded high-level observable features of the system and they must hold over the system's inputs and outputs. A detailed understanding of the program is required for the construction of metamorphic relations. When the program's functional behavior is unknown, there are no symmetries in the input data, and/or there is no evident system invariant; discovering metamorphic relations is challenging. Especially, it is difficult to systematically know for sure if a system has a metamorphic relation or if the metamorphic relation can discover faults, as it is dependent on expert knowledge.

The selection of metamorphic relations is another issue that testers face. It is inefficient and time-consuming to consider all the metamorphic relations and very difficult to determine which relations can find the most faults, especially in regression testing, where all the test suites must be run over and over again to discover unintentional faults.

The properties that must hold for different inputs and outputs are called metamorphic relations. Murphy et al[12] identified several metamorphic properties that can be considered for mathematical functions, as shown in Table 1. The table identifies several possible ways an input to a function can be modified, and, for a correct function, a correct metamorphic relation should predict the changes in the output from the changes in the input. For example, for a simple function that adds two numbers, an additive metamorphic relation will predict that adding a constant will add the same constant to the output. Therefore, Table 1 only suggests the types of input changes that can be done on mathematical function; however, how that change of the input implies a change in the output is only specific to the function. The metamorphic relations are named after the nature of that change of the input.

For this reason, it is said that a failure in metamorphic testing detects faults, but functions and software that pass are not guaranteed to be free of faults. For example, a mathematical function that adds two numbers might have the fault that it increments the output by one; an additive metamorphic relation will let the function pass for a change in input that is incremented by one, and the fault is never revealed.

Recent advances in machine learning (ML) enabled the automation of a range of software engineering tasks, including software testing. ML is primarily a set of algorithms for modeling and analyzing data. Mohri et al[13] define machine learning algorithms as data-driven approaches. Furthermore, it blends computer science's core ideas with principles from statistics, probability, and optimization. ML is centered on computer learning, according to Shalev-Shwartz and Ben-David[14]; hence, algorithmic notions are fundamental. The application of machine learning to software testing difficulties is a relatively young field of study. In the last two decades, a lot of scholars in this field have published their findings.[15] Different machine learning methods were investigated and applied to automate software testing in these papers. As a result, prediction analysis for the software under test may be utilized to automate the identification, generation, and selection of metamorphic relations.

There are two categories of examples in a machine learning algorithm. They are referred to as a "training set" and a "test set". The training set is used to train a predictive model using machine learning methods, whereas the test set is used to assess the prediction performance. The dataset's examples may or may not be labeled (i.e., the feature values correspond to a label). If the labels are binary and the task is classification, this task is specifically called a binary classification problem. In this study, two types of machine learning algorithms for automating the metamorphic relations identification process are examined. Features are a collection of input data qualities that determine the data's nature; each example in the data has a value for each feature. In supervised learning, the labels for the features are required in contrast to unsupervised learning where the labels are not required.

Barr et al[16] conducted a survey on test oracles. Their study on test oracles up to 2014 is comprehensively summarized in their survey. They divide test oracles into four types: those done manually, those automatically derived from development artifacts, those that represent the inherent properties of all programs, and those that require a person-in-the-loop to evaluate test results. Because machine learning approaches learn from project products rather than humans, they fall into the derived category. Some of the earliest attempts to derive oracles using machine learning are described.

TABLE 1 Metamorphic relations

Relation	Change made to the input
Additive	Add or subtract a constant
Multiplicative	Multiply by a constant
Permutative	Randomly permute the elements
Invertive	Take the inverse of each element
Inclusive	Add a new element
Exclusive	Remove an element
Compositional	Combining two or more inputs

Durelli et al[15] conducted thorough mapping research on the use of machine learning in software testing. Their focus is broader; however, they mentioned that machine learning has been used to aid in the creation of test oracles.

The research of Afonso et al.[17] focuses on the use of machine learning in oracle construction. Metamorphic test verdicts and expected output were used to categorize the program. Researchers also reported that the type of machine learning known as supervised learning is the most widely used. When it comes to machine learning techniques for software testing, artificial neural networks are the most often used algorithm.

The authors in Segura et al.[37] review several aspects of metamorphic testing. However, in this review, we focus solely on metamorphic relation (MR) automation. The authors in subsection 4.2 summarized six different approaches to MR automation. Our review is distinguished in that it provides a higher understanding of the objectives of MR automation. Also by including a higher number of articles (19 compared to 6) and more recent ones, it allows for a more detailed study and understanding of the topic. Our article also answers several research questions that are not tackled in the 2016 study.

Our study is unique in that it focuses solely on the automation of metamorphic relationships. This narrow emphasis allows for a more in-depth examination of this subject topic, which is not possible with larger surveys and mapping studies.

3 | RESEARCH METHODOLOGY

Our goal in this research is to investigate how researchers have automated the effort of creating and selecting metamorphic relations. We aimed to comprehend their techniques, findings, and conclusions. A secondary study (also known as a systematic literature review) was conducted to accomplish this goal. The approach is described in this section. The number of papers retrieved at each stage of the process is depicted in Figure 1.

In this SLR, we aim to investigate in detail the current techniques of metamorphic relation automation. Table 2 presents the research questions and the objectives of the questions.

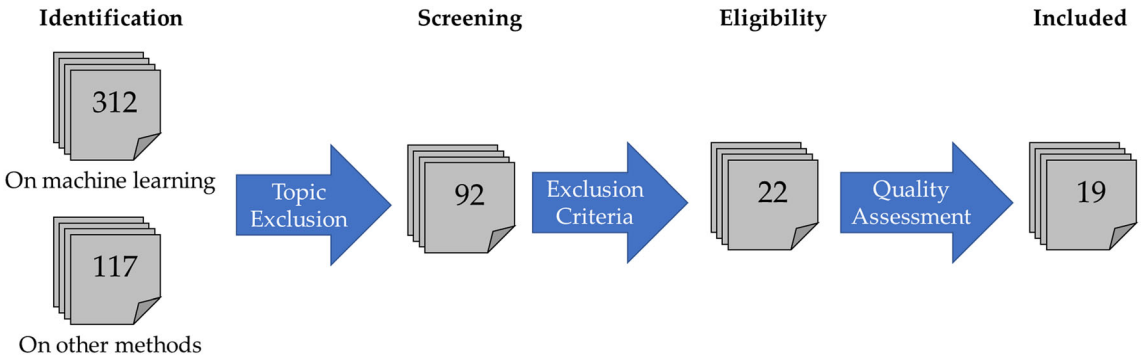

FIGURE 1 The number of papers at each stage of the methodology. SLR, systematic literature review

TABLE 2 Research questions and corresponding objectives

No.	Research questions	Objectives
RQ1	What are the objectives of metamorphic relation automation?	Whether it is selection, generation, or detection
RQ2	What are the techniques, approaches, or models used?	Understands how metamorphic relations are automated while highlighting trends and general procedures followed
RQ3	What are the prerequisites or input necessary for the automation method?	Highlights the nature of the input necessary to develop the automation tool (for example, data sets or human inputs)
RQ4	What are the types of metamorphic relations detected, generated, or selected in the software?	Because metamorphic relations are high-level property of the software, they can be any kind of relation (e.g., mathematical).
RQ5	What are the datasets used in the evaluation?	The datasets used for evaluation might skew the results.
RQ6	What are the evaluation metrics?	Many researchers opt for different kind of metrics depending on what they think best represent their tool.
RQ7	What are the challenges in the automation of metamorphic testing?	This is a qualitative question to capture the difficulties of metamorphic relation automation.

RQ1 to RQ4 help us understand how researchers automated metamorphic relations by understanding the features extracted, the techniques implemented, and the datasets used (or human input, if any). RQ5 and RQ6 review the evaluation procedures followed. RQ7 highlights the challenges faced by researchers in metamorphic relation automation.

3.1 | Primary study selection

A comprehensive search was conducted utilizing the following databases to discover the initial studies for consideration: IEEE Xplore, ACM Digital Library, Science Direct, and Scopus. By integrating key phrases related to automated metamorphic relation formation and selection, two search strings were constructed. The search phrases were chosen to be as broad as possible in order to find as many publications as conceivable, which we would then filter by relevance. The first search string (1) was intended to capture the application of machine learning methods in the development and selection of metamorphic relations. The second search string (2) sought to comprehend all automated metamorphic relation formation and selection procedures.

1. ("metamorphic testing" OR "metamorphic relations" OR "metamorphic relation") AND ("machine learning" OR "artificial intelligence" OR "deep learning" OR "neural network")
2. ("metamorphic relations" OR "metamorphic relation") AND ("detection" OR "generation" OR "selection") AND ("automatic" OR "automated")

We executed a two-step filtering procedure to exclude the irrelevant items from the original search, which yielded 429 articles. To begin, we manually examined all of the titles and weeded out any results that were unrelated to our research. The majority of the papers removed in this step were about using metamorphic testing to test machine learning algorithms, but our domain is applying machine learning to metamorphic testing, specifically metamorphic relations, and automatic metamorphic relations selection and generation, therefore, only 72 articles remained after this step. Afterward, the exclusion criteria detailed in Table 3 were applied, and then, 22 publications remained.

After that, we applied quality assessment of the papers based on eight quality measures defined in Table 4. After applying the quality assessment process, we ruled out three papers. The details are provided in Table A1. Figure 2 shows the distribution of the quality assessment scores of the selected papers.

Table B1 shows the final 19 papers included in the systematic literature review. Figure 3 depicts the distribution of research publications by publication year. This figure shows that several papers have been published recently and therefore, this study is timely research.

TABLE 3 Exclusion criteria

ID	Exclusion criteria
1	The study is not related to the current SLR context
2	The study is not in English
3	The study is a duplicate
4	The study is not available
5	The study is a secondary study

Abbreviation: SLR, systematic literature review.

TABLE 4 Quality assessment metrics

No.	The quality assessment metrics
QA1	Is the aim of the study defined well?
QA2	Is the experimental part defined well?
QA3	Is the methodology explained and defined well?
QA4	Are the threats to validity and weaknesses discussed?
QA5	Is the scope of the solution defined well?
QA6	Are the limitations of the work discussed?
QA7	Is the implementation of the ML algorithms discussed?
QA8	Are the negative findings presented?

Abbreviation: ML, machine learning.

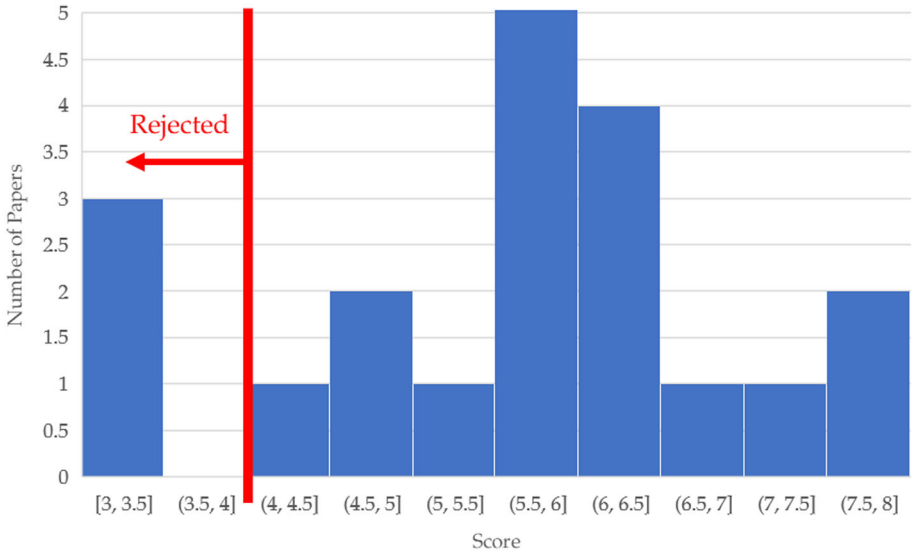

FIGURE 2 Graph of quality score of the articles. Articles with a score <4 were rejected.

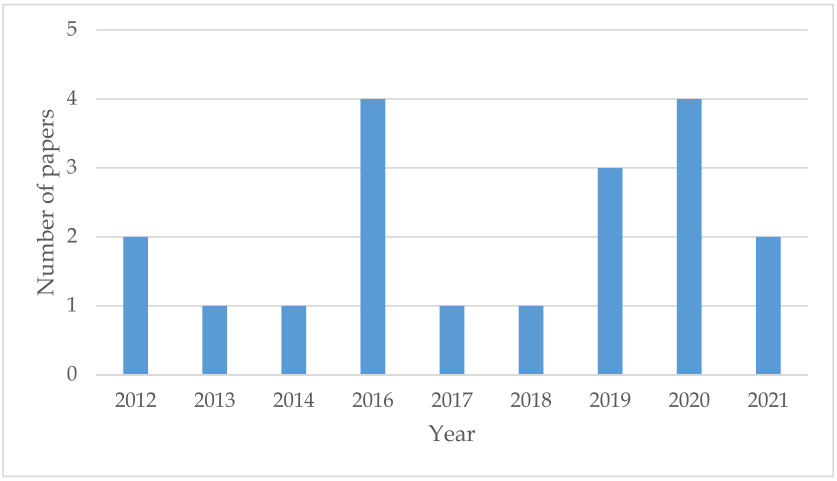

FIGURE 3 Distribution of publications per year

4 | RESULTS

After the data extraction process, our detailed findings are presented in this section. Each research question is answered and discussed. RQ3 is divided into two sub-questions; where the former is for machine learning approaches, and the latter is for other methods.

4.1 | RQ1: What are the objectives of metamorphic relation automation?

Three main objectives in MR automation have been found in the literature, which are detection, generation, and selection of MRs. Identification of MRs, which is also referred to as detection, refers to verifying if the system under test satisfies a certain set of pre-defined MRs. Generation of MRs, which is also referred to as inference, refers to the creation or extraction of new MRs from the system under test. Selection of MRs refers to finding the optimal set of correct metamorphic relations among the set of all defined metamorphic relations. Selection is rarely performed alone, but rather it is performed as a filtration or refinement process that comes after a set of MRs is generated. During our investigation in answering this RQ, we observed that ML approaches focused on different objectives than the rest. Such difference is depicted more clearly in Figures 4 and 5. Figures 4 and 5 show the distribution of objectives among the studies involving ML approaches and other approaches, respectively.

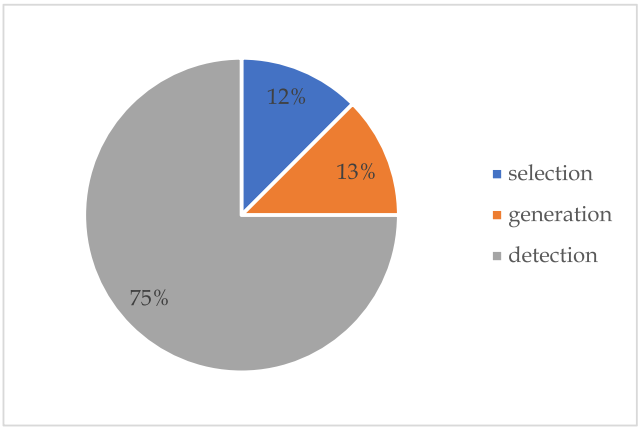

FIGURE 4 Objectives of metamorphic relation automation in ML approaches. ML, machine learning

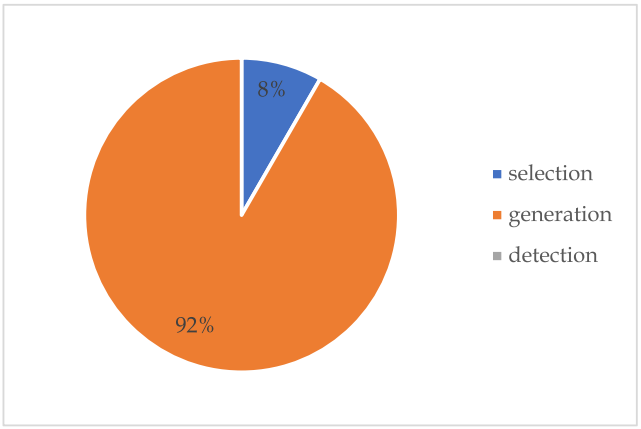

FIGURE 5 Objectives of metamorphic relation automation in other approaches

4.2 | RQ2: What are the techniques, approaches, or models used?

The techniques or models depend on the objective of the automation. For example, in ML approaches, the most common objective was detection. In detection, a function or a method is classified as whether it has a particular MR or not (e.g., Does function-1 have metamorphic relation A?). This is a binary class classification problem, and most of the literature employs support vector machines (SVM) to solve it. Another approach is the multi-label classification where each function is identified to have more than one MR (e.g., Function-1 has the metamorphic relations A, B, and C).

In selection, the ML model has to learn from previous test executions in which MRs were most likely to find faults. Therefore, researchers have opted for reinforcement learning, for example, a contextual bandit-based reinforcement algorithm was proposed in Spieker and Gotlieb,[29] where the model learns from previous test executions and selects the subset of MRs that is most likely to find faults.

Other approaches include MR composition approaches, category choice approaches, and search-based approaches. These approaches are discussed in detail in Sections 5.2 to 5.5.

4.3 | RQ3: What are the prerequisites or inputs necessary for the automation method?

4.3.1 | RQ3.1: What are the inputs for ML approaches?

ML approaches are data-driven; hence, they require a training dataset. In detection, most methods extract features from the control flow graph (CFG), where the features are node features and path features. The features, in this case, represent the "signature" of the function or method,

and the labels are whether this function satisfies a MR or not. Some methods include the node and path feature number and value change path. Another method applied graph kernels on CFG and program dependence graph (PDG), particularly random walk and graphlets. Another method applied a bag of words to extract textual features from program documentation.

In selection, one article that implemented reinforced learning based on the contextual bandit was found. It only accepts a collection of MRs, the system being tested, a collection of test cases, and a user-defined value based on the maximum number of iterations to execute as inputs.[29]

4.3.2 | RQ3.2: For methods other than ML, what are the inputs to the automation tool?

Inputs to the automation tools differ according to the nature of the system under test. For mathematical and scientific applications, the input is mostly several executions of the functions in the system with a large set of inputs. These inputs can be random numerical inputs or specific inputs designed according to a certain strategy. In addition, many studies use an initial set of MRs that are manually defined as an input or a prerequisite to their approach. For instance, one of the studies was on generating new MRs by combining two or more pre-defined ones.[23]

4.4 | RQ4: What are the types of MRs detected, generated, or selected in the software?

The type of MRs mostly depends on the objective of automation. If the objective is to select a subset of existing MRs or to filter and refine the generated MRs, then the MR can be of any type. For instance, one study used the results of ML analysis to create and refine MRs that are related to the testing simulation of a light simulation software.[28] On the other hand, if the objective is to generate new MRs from scratch, then the type of MRs is mostly purely mathematical. Examples of those types are found in studies of Zhang et al. and Zhang et al.[18,31]

4.5 | RQ5: What are the datasets used in evaluation?

The datasets used to evaluate the approaches are the programs or functions that are being tested using metamorphic testing techniques using the identified, generated, or selected MRs. A lot of the studies are performed on mathematical applications. These studies usually use a set of functions from mathematical libraries and software to evaluate their approaches, such as Apache Commons Math, NumPy, and MATLAB. For other studies that deal with other fields, datasets vary according to the application. Examples of the software applications used include a bioinformatics program called DNAPARS,[23] a light simulation software called ADDA,[28] a model transformation tool called PetriNet2PNML,[20] and software for elevator management from Orona company.[19]

4.6 | RQ6: What are the evaluation metrics used?

Evaluation metrics used in the studies are summarized in Table 5. Most studies evaluate the performance of their approach using metrics related to the number and correctness of the generated or identified MRs, such as the number of true positives out of all inferred relations. Furthermore, some studies evaluate their approach by performing mutation testing to check the performance and usefulness of the generated MRs for killing mutants of the system under test. These studies use the mutation score as the main evaluation metric. In addition, ML-based approaches use metrics related to ML such as receiver operating characteristic metric, balanced success rate, and area under the receiver operating characteristic curve.

4.7 | RQ7: What are the limitations and challenges in the automation of metamorphic testing?

Many studies acknowledged limitations and challenges to their approaches. These limitations can be categorized into mathematical-related, ML-related, and domain-related limitations.

Studies that deal with mathematical applications acknowledged that their approaches only work with numerical values as inputs; they cannot work with arrays or pointers, and they cannot work with changing data such as data obtained from databases and changing attributes of an object in object-oriented applications.

The main challenge in ML-based approaches is the lack of large labeled open-source datasets to train ML models. In addition, many studies that do not deal with pure mathematical applications acknowledged that their proposed approach works well in their domain but is not guaranteed to work on other programs or applications from other domains.

TABLE 5 Frequency of used evaluation metrics

Metric	Number of times used
Number of inferred MRs	4
Mutation score	4
True positives among all inferred MRs (TP)	2
False positives among all inferred MRs (FP)	3
Receiver Operating Characteristic metric (ROC)	1
Balanced Success Rate (BSR)	1
Area Under the Receiver Operating Characteristic Curve (AUC)	3
Hamming loss	1
Coverage	1
Average Precision	1
Accuracy	2
Time Consumption	1
Violation Rate	1
One-error	1
Ranking loss	1

5 | DISCUSSION

In this section, we discuss our findings based on the research questions and then systematically categorize our findings and what we have learned from the literature.

5.1 | General discussion

The main theme in the research articles analyzed is that most of the applications where MR automation is performed are mathematical or scientific software testing such as in Zhang et al. and Zhang et al.[18,31] except for some approaches where the proposed method can only be used in a specific domain such as in Ayerdi et al., Liu et al., and Ding et al.[19,23,28] This shows that a lot of attention is directed toward mathematical types of metamorphic relations in the research community.

It is worth noting that ML-based approaches focus on the detection of present MR, whereas other approaches focus on the generation of new MRs from scratch. Selection of a subset of manually identified MRs is rarely done in the literature as only two research articles have been found tackling this issue. Rather, most research studies perform selection, filtration, or refinement of automatically generated MRs.

MR discovery approaches found in the literature can be categorized into static and dynamic approaches. Static approaches involve static analysis of different forms of data related to the system under tests, such as analysis of CFG, PDG, and software documentation. Moreover, static approaches also include manual detection of MRs. Static approaches include ML-based approaches, pattern-based approaches, and category selection-based approaches. The main limitation with static approaches is that they require high-level knowledge of the system under test, which poses an issue of scalability as large-scale applications can be hard to analyze. On the other hand, dynamic approaches are executed dynamically, and the MRs are derived automatically. These approaches include search-based approaches and MRs composition-based approaches. Dynamic approaches overcome the limitations of static approaches, but they have their own limitations such as low accuracy, limited forms of MR expression as most of them are purely mathematical, and a lack of description of generated MRs.

On another level, all approaches found in the literature for the automatic generation or detection of MRs can be categorized into six main categories as shown in Table 6. The table shows each approach type with its prerequisites and the degree of automation involved in the approach. It can be seen that some approaches are considered manual as they involve a lot of manual tasks such as creating new MRs by combining other relations.[23] Other methods are considered semi-automatic as they involve some manual tasks. Although there are three approaches that are considered fully automatic, two of them require an initial set of MRs, so the only method that can be considered as a generation of MRs from scratch is the search-based approach.

TABLE 6 Categories of metamorphic relation inference

Approaches	Reference	Inputs	Objective and outputs	Automation degree	Advantages	Disadvantages
MR composition	[23]	Existing MRs	Generation of more complex MRs	Manual	- Produces complex relations - Reduces the number of MRs in testing	- Requires high knowledge of the system - Requires an initial set of MRs - Not scalable
Input domain-based category choice	[22]	Complete test frames (CTFs)	Generation of new MRs	Semi-automatic	- Applicable to a large scope of applications - Does not require initial MRs	- Partially requires testers' expertise - Does not utilize output domain
Input and output domain-based category choice	[21]	IO-CTFs	Generation of new MRs	Semi-automatic	- Improved by utilizing output domain - Applicable to a large scope of applications - Does not require initial MRs	- Partially requires testers' expertise
Machine learning-based MR detection	[24–28]	Existing MRs	Detection of MRs satisfied by the SUT	Fully automatic	- High degree of automation - Expert knowledge is not needed	- Require manually identified MRs - Mostly limited to mathematical and scientific software - Lack of high-quality datasets
Search-based MR inference	[18,31]	Multiple program executions	Generation of new MRs	Fully automatic	- Infers MRs from scratch - Does not require access to source code - Does not require initial MRs	- Produces invalid MRs if the implementation is faulty - Limited to mathematical applications
Selection approaches	[29,32]	Previous MT results	Selection of most useful MRs	Fully automatic	- Applicable to any SUT with identified MRs	- Require prior MT on all MRs - If new MRs are added, MR selection is recomputed

Abbreviations: MR, metamorphic relation; MT, metamorphic testing; SUT, system under test.

5.2 | MR generation approaches

5.2.1 | MR composition approach

A MR composition is an approach that combines two or more relations to build a new relation.[23] The idea of this approach is that all properties of the combined relations are embedded in the new relation. This allows for achieving the same effectiveness in testing with a smaller number of MRs. On the other hand, determining if two or more relations are compositable is mostly done manually. Furthermore, the order of the combined relations largely impacts the resulting relation. Thus, this approach has a very low automation degree and requires expert knowledge of the system under test, which signifies the scalability issue.

5.2.2 | Category choice approaches

Category choice is an approach that utilizes the software specifications in the identification of MRs. In short, input arguments and environmental conditions that have an impact on the software execution are referred to as categories. Choices are separate sections of each category that cover

the potential values in the category. METRIC[22] is an approach that utilizes categories and choices in the software specifications to identify MRs in the software without the need for initial MRs. As an improvement, METRIC+[21] was developed, which is a similar approach that utilizes information extracted from the output domain of the software in addition to the input domain, thus improving the effectiveness of the approach. The main advantage of the category choice approach is its high applicability unlike a lot of the other approaches that can only be applied to one domain. Another advantage to this approach is that it does not require any initial MRs in the identification process. On the other hand, this approach is considered semi-automatic as it still partially depends on the expert knowledge of the tester in MRs identification.

5.2.3 | Search-based approaches

Search-based MR inference approaches rely on multiple program executions to construct MRs in certain mathematical forms. For instance, Zhang et al[18] proposed a search-based approach to generate MRs in polynomial forms, whereas Zhang et al[31] proposed a method to generate and cleanse MRs in the form of equalities and inequalities. These approaches use searching algorithms to search for MRs among multiple program executions with several random inputs and their respective outputs. One major advantage of search-based approaches is that they generate MRs from scratch without the need for an initial set of MRs. Another advantage is that they are considered black-box approaches, which means they do not require access to the source code of the program. On the other hand, the main disadvantage of search-based approaches is that they are limited to mathematical applications. Furthermore, because they rely on program executions, they are very sensitive to the correctness of code implementation as any fault in the program will yield invalid MRs.

5.3 | MR detection approaches

MR detection was performed mainly using ML approaches. Most ML approaches use supervised classification (binary or multi-class classification). Supervised binary classification is used to classify relations as satisfied by the software or not, which is achieved by analyzing software-related data such as CFGs, PDGs, and software documentation along with an initial set of MRs, such as in Kanewala and Bieman, Kanewala et al., Nair et al., and Spieker and Gotlieb.[24,26,27,29] Another approach to identification is the multi-label classification problem where the model predicts all the MRs a particular function or method has, such as in Zhang et al.[25] Only one study was found on the use of reinforcement learning for MR selection.[29] The key strength of ML approaches is their high degree of automation, as they do not require any expert knowledge in MRs. However, the drawback of these approaches is the need for an initial set of MRs. In addition, the main challenge in using ML for MR identification is the scarcity of big and consistently labeled open-source software datasets that can be used for training ML models as acknowledged in Nair et al., and Ding and Zhang.[27,28]

5.4 | MR selection approaches

The main objective of selection approaches is to increase the efficiency of metamorphic testing by finding the most useful set of MRs to be given priority in testing. Srinivasan and Kanewala[36] proposed an approach for MR prioritization based on fault detection information and code coverage. In fault-based MR prioritization, previous test results are utilized to find the MRs that revealed the highest number of faults. In coverage-based MR prioritization, information from previous test results in terms of statement or branch coverage is used to determine which MRs obtained the highest coverage in the testing process. Spieker and Gotlieb[29] proposed a testing framework called adaptive metamorphic testing. This framework uses reinforcement learning with contextual bandits to select MRs that are more likely to reveal faults in the system under test (SUT). Based on the selected MRs, follow-up test cases can be generated. Both studies have shown that their approaches increase the efficiency and effectiveness of metamorphic testing (MT). The main advantage of these selection methods is their high applicability, as they can theoretically be applied to any system with a set of identified MRs. On the other hand, one drawback of these approaches is that they require the results of previous extensive metamorphic testing on all MRs. Another drawback is when the software changes introduce new MRs, the ordered list of prioritized MRs needs to be recomputed from scratch.

5.5 | Limitations and challenges

Although the automation of MR detection and generation is most effective in approaches that are based on pure mathematical or scientific functions, there are a few limitations to these approaches. The main limitation is that the input to the system under test must be a numerical value. This means that they cannot be used with applications that involve pointers, arrays, or sets of numerical values as acknowledged in Zhang et al.[18]

Another limitation is that these approaches work with functions and programs where the same input should return the same output. Therefore, they cannot be used for applications involving databases or object-oriented features such as modifying attributes of an object as acknowledged in Zhang et al.[18]

On the other hand, for approaches that do not necessarily involve pure mathematical MRs, the main limitation is that they are domain-specific. This means that they are developed for a certain application and there is no guarantee that they can work in other domains. For instance, a study used genetic algorithms to develop a technique to be used for cyber-physical applications and presented a case study of software for managing elevators.[19] The authors acknowledged that their results may not represent other cyber-physical systems from different domains. In another study, a MR composition approach was developed to test DNAPARS, a bioinformatics software.[23] The authors acknowledge that although DNAPARS is a typical program with oracle problems, their technique is not guaranteed to work with other types of programs.

For ML-based approaches used for MR identification, many studies such as Nair et al., and Ding and Zhang,[27,28] acknowledged that the main challenge is the lack of large and consistently labeled publicly available software datasets that can be used to train ML models sufficiently. Another challenge is that these approaches are limited to the detection of MRs rather than generating new ones from scratch.

MR selection approaches have been shown to be effective in the selection and prioritization of the most useful set of MRs. Nonetheless, they suffer from some limitations. First, they require the results of extensive metamorphic testing using all available MRs. Second, if a new MR or a set of MRs is added or discovered, the whole MR prioritization process needs to be recomputed to produce a new ordered list of the most useful MRs.

5.6 | Potential threats to validity

The main possible threat to validity is that posed by the choice of the keywords, however, we believe that all articles are related to the automation of MRs. They at least mention somewhere in the manuscript the keywords we have chosen. Another threat to validity is the filtering process, because most of the articles were filtered only on the basis of the title, we might have filtered out articles that are related to the study. However, we validated our study by examining all the references from the final set of papers and found that we have not missed any articles. Another threat to validity is the quality assessment process, which might lead to some articles failing even though they should have passed, but because only three articles failed to pass, this will not affect the findings of this paper. The last threat to validity is the data extraction process, which is prone to human error and misunderstanding.

6 | CONCLUSION AND FUTURE WORK

This paper presents a SLR of approaches to improve automation in MRs acquisition for metamorphic testing purposes. There are three main objectives that have been found in the literature regarding MR automation. The first is detection, which refers to checking if the system under test satisfies certain MRs or not. The second is the generation, which refers to the inference or creation of new MRs. The third objective is selection, which refers to finding the most efficient MRs out of a large set.

The studies show that automation of MR generation is most effective in mathematical and scientific applications. These types of applications mostly deal with pure mathematical MRs. Other approaches can be effective in their specific domains with any type of MRs; however, these are not guaranteed to work with other domains.

In addition, the review indicates that the approaches vary in their degree of automation regarding the generation and detection of MRs. For instance, many approaches require manual analysis and testers' expertise such as MR composition and category-choice approaches. Others rely on existing manually defined relations as input such as most ML-based approaches. These approaches are considered semi-automatic. On the other hand, search-based methods can be considered fully automatic as they automatically generate MRs from scratch by analyzing the results of multiple runs of the system under test with a large set of inputs, but they are limited to mathematical applications. Furthermore, some approaches involve analysis of different forms of software-related information such as CFGs, PDGs, software documentation, and UML diagrams as well as an initial set of MRs. Those approaches are mostly ML-based. Other approaches involve analysis of several executions of the software functions with randomly or specifically generated inputs such as category choice and search-based approaches. Selection methods have been shown to improve the efficiency and effectiveness of metamorphic testing while being applicable to a wide range of fields.

ML techniques especially supervised binary classification, have been used mainly to identify if the system under test satisfies a set of MRs rather than generating new ones from scratch. For ML-based approaches, the main limitation is the lack of adequate datasets for training the models. However, the use of ML in the detection and generation of MRs has great potential, but it has not been thoroughly researched yet as evident from the small number of publications in this field. For instance, deep learning has not been utilized in the literature despite its great effectiveness and high automation degree. Furthermore, interpretable ML would be significantly useful in understanding the process of generation and detection of MRs instead of using classical ML as a black-box technique.

In general, this review signifies the problem of applicability in MR automation, as it has been shown that automatic generation of new MRs for an arbitrary system is difficult. It also indicates another main obstacle which is the requirement of non-trivial domain knowledge in the MR generation process. It is clear that the field of automation of MRs is largely unexplored.

ACKNOWLEDGMENTS

This publication is supported in part by grant NPRP12C-33905-SP-66 and from the Qatar National Research Fund. The findings achieved herein are solely the responsibility of the authors. Open Access funding provided by the Qatar National Library.

DATA AVAILABILITY STATEMENT

Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

ORCID

Abdullah Elkawakjy https://orcid.org/0000-0002-3381-7098
Cagatay Catal https://orcid.org/0000-0003-0959-2930

REFERENCES

1. Planning S. *The economic impacts of inadequate infrastructure for software testing*. National Institute of Standards and Technology; 2002.
2. Myers GJ. *The art of software testing*. John Wiley & Sons; 2006.
3. Chen TY, Tse TH, Zhou ZQ. Fault-based testing without the need of oracles. *Inf Softw Technol*. 2003;45(1):1-9. doi:10.1016/S0950-5849(02)00129-5
4. Weyuker EJ. On testing non-testable programs. *Comput J*. 1982;25(4):465-470. doi:10.1093/comjnl/25.4.465
5. Kelly D, Sanders R. The challenge of testing scientific software. In: *Proceedings of the 3rd annual conference of the Association for Software Testing (CAST 2008: Beyond the Boundaries)*. Citeseer; 2008:30-36.
6. Zhou ZQ, Huang D, Tse T, Yang Z, Huang H, Chen T. Metamorphic testing and its applications. In: *Proceedings of the 8th International Symposium on Future Software Technology (ISFST 2004)*. Xian, China: Software Engineers Association; 2004:346-351.
7. Shahri MP, Srinivasan M, Reynolds G, Bimczok D, Kahanda I, Kanewala U. Metamorphic testing for quality assurance of protein function prediction tools. In: *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*. IEEE; 2019:140-148.
8. Zhou ZQ, Xiang S, Chen TY. Metamorphic testing for software quality assessment: a study of search engines. *IEEE Trans Softw Eng*. 2015;42(3):264-284. doi:10.1109/TSE.2015.2478001
9. Kitchenham B, Brereton OP, Budgen D, Turner M, Bailey J, Linkman S. Systematic literature reviews in software engineering–a systematic literature review. *Inf Softw Technol*. 2009;51(1):7-15. doi:10.1016/j.infsof.2008.09.009
10. Tummers J, Kassahun A, Tekinerdogan B. Obstacles and features of farm management information systems: a systematic literature review. *Comput Electron Agri*. 2019;157:189-204. doi:10.1016/j.compag.2018.12.044
11. Chen TY, Cheung SC, Yiu SM. Metamorphic testing: a new approach for generating next test cases. *arXiv preprint arXiv:2002.12543*, 2020.
12. Murphy C, Kaiser GE, Hu L. Properties of machine learning applications for use in metamorphic testing. 2008.
13. Mohri M, Rostamizadeh A, Talwalkar A. *Foundations of machine learning*. MIT press; 2018.
14. Shalev-Shwartz S, Ben-David S. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge university press; 2014.
15. Durelli VH, Durelli RS, Borges SS, et al. Machine learning applied to software testing: a systematic mapping study. *IEEE Trans Reliab*. 2019;68(3):1189-1212. doi:10.1109/TR.2019.2892517
16. Barr ET, Harman M, McMinn P, Shahbaz M, Yoo S. The oracle problem in software testing: a survey. *IEEE Trans Softw Eng*. 2014;41(5):507-525. doi:10.1109/TSE.2014.2372785
17. Fontes A, Gay G. Using machine learning to generate test oracles: a systematic literature review in *Proceedings of the 1st International Workshop on Test Oracles*, 2021, pp. 1-10. [37] Segura, Sergio, et al. A survey on metamorphic testing. *IEEE Trans Softw Eng*. 2016;42(9):805-824.
18. Zhang J, Chen J, Hao D, et al. Search-based inference of polynomial metamorphic relations. In: *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*; 2014:701-712.
19. Ayerdi J, Terragni V, Arrieta A, Tonella P, Sagardui G, Arratibel M. Generating metamorphic relations for cyber-physical systems with genetic programming: an industrial case study. In: *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*; 2021:1264-1274.
20. Troya J, Segura S, Ruiz-Cortés A. Automated inference of likely metamorphic relations for model transformations. *J Syst Softw*. 2018;136:188-208. doi:10.1016/j.jss.2017.05.043
21. Sun C-A, Fu A, Poon P-L, Xie X, Liu H, Chen TY. Metric+: a metamorphic relation identification technique based on input plus output domains. *IEEE Trans Softw Eng*. 2019;47(9):1764-1785. doi:10.1109/TSE.2019.2934848
22. Chen TY, Poon P-L, Xie X. METRIC: METamorphic relation identification based on the category-choice framework. *J Syst Softw*. 2016;116:177-190. doi:10.1016/j.jss.2015.07.037
23. Liu H, Liu X, Chen TY. A new method for constructing metamorphic relations. In: *2012 12th International Conference on Quality Software*. IEEE; 2012:59-68.
24. Kanewala U, Bieman JM. Using machine learning techniques to detect metamorphic relations for programs without test oracles. In: *2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE; 2013:1-10.
25. Zhang P, Zhou X, Pelliccione P, Leung H. RBF-MLMR: a multi-label metamorphic relation prediction approach using RBF neural network. *IEEE Acc*. 2017;5:21791-21805. doi:10.1109/ACCESS.2017.2758790
26. Kanewala U, Bieman JM, Ben-Hur A. Predicting metamorphic relations for testing scientific software: a machine learning approach using graph kernels. *Softw Test Verif Reliab*. 2016;26(3):245-269. doi:10.1002/stvr.1594

27. Nair A, Meinke K, Eldh S. Leveraging mutants for automatic prediction of metamorphic relations using machine learning. In: *Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation*; 2019:1-6.
28. Ding J, Zhang D. A Machine Learning Approach for Developing Test Oracles for Testing Scientific Software. In: *SEKE*; 2016:390-395.
29. Spieker H, Gotlieb A. Adaptive metamorphic testing with contextual bandits. *J Syst Softw*. 2020;165:110574. doi:10.1016/j.jss.2020.110574
30. Rahman K, Kahanda I, Kanewala U. MRpredT: Using text mining for metamorphic relation prediction. In: *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*; 2020:420-424.
31. Zhang B, Zhang H, Chen J, Hao D, Moscato P. Automatic discovery and cleansing of numerical metamorphic relations. In: *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE; 2019:235-245.
32. Li J, Liu L, Zhang P. Tabular-expression-based method for constructing metamorphic relations. *Softw Practice Exp*. 2020;50(8):1345-1380.
33. Singh G. An automated metamorphic testing technique for designing effective metamorphic relations. In: *International Conference on Contemporary Computing*. Springer; 2012:152-163.
34. Zhang X, Yu L, Hou X. A method of metamorphic relations constructing for object-oriented software testing. In: *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. IEEE; 2016:399-406.
35. Li M, Wang L, Yan S, Yang X. Metamorphic relation generation for physics burnup program testing. *Int J Perform Eng*. 2020;16(2):297-306. doi:10.23940/ijpe.20.02.p12.297306
36. Srinivasan M, Kanewala U. Metamorphic relation prioritization for effective regression testing. *arXiv preprint arXiv:2109.09798*, 2021.
37. Segura S, Fraser G, Sanchez AB, Ruiz-Cortés A. A survey on metamorphic testing. *IEEE Trans Softw Eng*. 2016;42(9):805-824. doi:10.1109/TSE.2016.2532875
38. Mayer J, Guderlei R. On random testing of image processing applications. In: *2006 Sixth International Conference on Quality Software (QSIC'06)*. IEEE; 2006:85-92.
39. Sim KY, Pao WKS, Lin C. Metamorphic testing using geometric interrogation technique and its application. *ECTI Trans Comput Inf Technol (ECTI-CIT)*. 2005;1(2):91-95. doi:10.37936/ecti-cit.200512.51837
40. Chen T, Yueh F, Kuo HL, Wang S. Conformance testing of network simulators based on metamorphic testing technique. In: *Formal Techniques For Distributed Systems*. Berlin, Heidelberg: Springer; 2009:243-248.
41. Núñez A, Hierons R. A methodology for validating cloud models using metamorphic testing. *Annals Telecommun Annales Des télécommunications*. 2015;70(3):127-135. doi:10.1007/s12243-014-0442-7

How to cite this article: Altamimi E, Elkawakjy A, Catal C. Metamorphic relation automation: Rationale, challenges, and solution directions. *J Softw Evol Proc*. 2023;35(1):e2509. doi:10.1002/smr.2509

APPENDIX A

TABLE A1 Quality assessment details

Article title	q1	q2	q3	q4	q5	q6	q7	q8	Total
Search-based inference of polynomial metamorphic relations	1	1	1	1	0.5	0.5	1	0	6
Generating metamorphic relations for cyber-physical systems with genetic programming: An industrial case study	1	0.5	0.5	1	1	1	0.5	1	6.5
Metamorphic relation prioritization for effective regression testing	1	1	0.5	1	0.5	0.5	0.5	1	6
Metamorphic relation generation for physics burnup program testing.	1	1	1	0	0.5	0	1	0	4.5
Automated inference of likely metamorphic relations for model transformations	1	1	0.5	1	1	1	1	1	7.5
METRIC+: A metamorphic relation identification technique based on input plus output domains	1	1	1	1	0.5	1	0.5	0	6
METRIC: METamorphic Relation Identification based on the Category-choice framework	1	1	1	1	1	1	0.5	0.5	7
A method of metamorphic relations constructing for object-oriented software testing	1	1	1	0	0.5	1	0.5	0	5
A genetic algorithm-based approach for composite metamorphic relations construction	1	0.5	1	0	0.5	0	0.5	0	3.5
A new method for constructing metamorphic relations	1	1	1	0.5	0.5	1	1	0	6
An automated metamorphic testing technique for designing effective metamorphic relations	1	1	0.5	0.5	0.5	1	0.5	1	6
Tabular-expression-based method for constructing metamorphic relations	1	1	0.5	0.5	1	0.5	1	0.5	6
M.R. Hunter: Hunting for metamorphic relations by puzzle solving	0.5	0.5	0.5	0	1	0.5	0.5	0	3.5

TABLE A1 (Continued)

Article title	q1	q2	q3	q4	q5	q6	q7	q8	Total
Automatic discovery and cleansing of numerical metamorphic relations	1	1	1	0	0.5	0.5	1	0	5
Using machine learning techniques to detect metamorphic relations for programs without test oracles	1	1	1	1	0.5	0.5	1	0	6
RBF-MLMR: A multi-label metamorphic relation prediction approach using rbf neural network	1	1	1	1	1	1	1	1	8
Predicting metamorphic relations for testing scientific software a machine learning approach using graph kernels	1	1	0	0.5	0.5	1	0.5	1	5.5
MRpredT: Using text mining for metamorphic relation	1	1	1	0.5	0.5	1	0.5	1	6.5
Leveraging mutants for automatic prediction of metamorphic relations using machine learning	1	1	1	0.5	0.5	1	0.5	1	6.5
A machine learning approach for developing test oracles for testing scientific software	1	1	1	0.5	1	0.5	1	0.5	6.5
Adaptive metamorphic testing with contextual bandits	1	1	1	1	1	1	1	1	8
Automated identification of metamorphic test scenarios for an ocean-modeling application	0	0.5	0.5	0	1	0.5	0	0	3

APPENDIX B

TABLE B1 Final set of selected articles

Reference	Title	Year
[36]	Metamorphic relation prioritization for effective regression testing	2021
[19]	Generating metamorphic relations for cyber-physical systems with genetic programming: An industrial case study	2021
[29]	Adaptive metamorphic testing with contextual bandits	2020
[30]	MRpredT: Using text mining for metamorphic relation	2020
[32]	Tabular expression based method for constructing metamorphic relations	2020
[35]	Metamorphic relation generation for physics burnup program testing	2020
[27]	Leveraging mutants for automatic prediction of metamorphic relations using machine learning	2019
[31]	Automatic discovery and cleansing of numerical metamorphic relations	2019
[21]	Metric+: A metamorphic relation identification technique based on input plus output domains	2019
[20]	Automated inference of likely metamorphic relations for model transformations	2018
[25]	RBF-MLMR: A multi-label metamorphic relation prediction approach using RBF neural network	2017
[22]	METRIC: METamorphic Relation Identification based on the Category-choice framework	2016
[26]	Predicting metamorphic relations for testing scientific software: a machine learning approach using graph kernels	2016
[28]	A machine learning approach for developing test oracles for testing scientific software	2016
[34]	A method of metamorphic relations constructing for object-oriented software testing	2016
[18]	Search-based inference of polynomial metamorphic relations	2014
[24]	Using machine learning techniques to detect metamorphic relations for programs without test oracles	2013
[23]	A new method for constructing metamorphic relations	2012
[33]	An automated metamorphic testing technique for designing effective metamorphic relations	2012