

Metamorphic Testing of High-Utility Itemset Mining

Tzung-Pei Hong^{1,2}, Rang Lee², Bay Vo³, Shu-Min Li²

¹National University of Kaohsiung, Kaohsiung, 811, Taiwan

²National Sun Yat-Sen University, Kaohsiung, 804, Taiwan

³HUTECH University, Ho Chi Minh City, Vietnam

tp hong@nuk.edu.tw, 4a0g0902@stust.edu.tw,

vd.bay@hutech.edu.vn, smli@cse.nsysu.edu.tw

Abstract. Because the internet and storage technology are developing rapidly, many new applications focusing on big data have emerged. Finding meaningful knowledge from large amounts of data has thus become an important issue. In the past, utility mining was proposed to find significant itemsets based on their potential benefits. Because the utility-mining process is complex, judging the correctness of its programs is very time-consuming and crucial to its applications. In this paper, we adopt a famous software-testing technique called metamorphic testing to verify the correctness of a utility-mining program. The metamorphic testing uses metamorphic relations as an aid to test the correctness. When input conditions are changed, the program is correct with more confidence if the corresponding outputs follow the rules, but if they do not, errors certainly exist in the program. We design fourteen metamorphic relations for utility mining to verify the correctness of its programs. These relations are designed based on two aspects: changing the minimum utility threshold and modifying the content of a transaction database. In the experiments, we use the mutation score to evaluate the effectiveness of the metamorphic relations.

Keywords: data mining, utility mining, metamorphic testing, metamorphic relation, software testing.

1 Introduction

With the rapid development of information applications, the volume of data is continually increasing. People are trying to extract useful information from vast databases to formulate strategies. As a result, various data mining techniques have been proposed to unearth valuable insights. These mined insights allow decision-makers to make informed and efficient decisions to adapt to the ever-changing business landscape.

Association-rule mining [1] only considers whether an item is present in a transaction or not. However, frequent itemsets identified in association-rule mining may cause little profit since their prices may be low. Yao et al. thus proposed the utility-mining problem to meet the business requirement [2].

However, companies typically do not hire specialized teams to extract useful information from their databases as it increases operational costs. Therefore, with the advancement of cloud computing, many companies have started outsourcing data mining tasks to third-party service providers. Thus, it cannot be guaranteed that the returned

mining results from outsourcing are entirely correct. Hence, the accuracy and completeness of the returned itemsets are crucial for companies. Incorrect item sets can lead to erroneous decisions and significant losses.

In order to deal with this problem, we adopt a software-testing technique called metamorphic testing in this paper, which was proposed by Chen et al. [3]. The testing method tests whether a program is correct by defining some metamorphic relations (MRs). A metamorphic relation is composed of an input relation and an output relation. An input relation is used to change an original test case into another one. An output relation is used to check whether the outputs from both the original and the changed test cases will follow the variation property caused by the input change. If the output results of the two cases satisfy the output relation, then the program has a high probability of being correct. On the contrary, the program clearly has errors.

Currently, metamorphic testing is mostly applied in machine learning, with only a few applications in data mining. In this paper, we will design several metamorphic relations and will use these relations to verify high-utility mining programs. In the experiments, we will use the mutation score to evaluate the effectiveness of the metamorphic relations.

2 Related Works

In this section, we review some works related to the paper. They are high-utility mining and metamorphic testing.

2.1 High-Utility Mining

Most data mining methods determine the importance of an itemset in a transaction database based on the frequency or occurrence of the itemset in the transactions. However, for a marketer, the most important issue is to identify the combinations of items that can contribute the most to the overall profit. Frequently occurring itemsets often represent only a small portion of the total profit, while those infrequent itemsets with high unit prices can often generate high profits. Therefore, Yao et al. proposed a research topic called "high-utility itemset mining" [4] to evaluate the actual utility of an item or itemset in a transaction database. The main concept is that the utility value of an item in a transaction record can be obtained by multiplying the number of purchases of the item in the record by its unit utility value. Suppose the total utility of an itemset in the whole database is high enough, exceeding a certain threshold. In that case, it is considered an important itemset and becomes a high-utility itemset. High-utility mining does not have downward closure, so its complexity is more challenging than frequent pattern mining.

Yao et al. also proposed a mining algorithm called MEU (Mining using Expected Utility) [5] to obtain all high-utility itemsets. Although this method can be applied to the high-utility mining problem, it first needs to generate all possible combinations for all different items in the database and then scan the database to obtain the actual utility of each combination. To reduce the search space for finding high-utility itemsets, Liu

et al. proposed a two-phase algorithm [6] in 2005, which also introduced a processing mode that utilizes the upper bound of utility values and has downward closure. The first stage applies the upper bound pattern to find a high transaction-weighted utilization itemset that satisfies the minimum utility threshold. Then, in the second stage, the database is rescanned to update the actual utility contributions of all high transaction-weighted utilization itemsets, thus obtaining all high-utility itemsets. Subsequently, many scholars have proposed different methods to speed up its execution efficiency. Ma et al. proposed an algorithm with PHUI-Tree in 2009 [15]. In 2020, Hong et al. proposed an algorithm to mine high temporal fuzzy utility itemsets with a tree structure [14]. J. Vo et al. proposed an algorithm to mine correlated high-utility itemsets in one phase in 2020 [28]. In 2021, Yun et al. proposed utility-oriented data analytics for transaction modifications in the internet of things [25]. Lin et al. proposed large-scale closed high-utility itemset mining [27]. Pushp et al. proposed an algorithm to mine high-utility itemsets from incremental datasets in 2021 [17]. In the same year, Pushp et al. proposed an algorithm to mine high-utility itemsets with negative utility values from incremental datasets [18]. In 2022, Hong et al. proposed an incremental fuzzy utility mining algorithm with a tree structure [19]. In 2023, Qu et al. proposed an algorithm using prefix trees and utility vectors to mine high-utility itemsets [20]. Kim et al. proposed an efficient method for mining high-utility occupancy patterns based on an indexed list structure [26].

2.2 Metamorphic Testing

To test whether software contains errors, Chen et al. proposed an approach called metamorphic testing. In metamorphic testing, a crucial feature is the ability to formulate specific metamorphic relations for different problem domains. The main concept involves executing test cases before and after metamorphosis in the program, obtaining corresponding outputs for each, and then comparing the relationship between these two outputs to the defined output relations in the metamorphic relations. If they match, there is more confidence that the program is correct; if not, there may be errors in the program.

In metamorphic testing, designing effective metamorphic relations is crucial for achieving good testing results. Segura et al. provided several perspectives on how to design effective metamorphic relations: (1) having good background knowledge in the relevant domain for formulating metamorphic relations, (2) maximizing differences between the follow-up input (after metamorphosis) and source input (before metamorphosis) in metamorphic relations, (3) using metamorphic relations for specific parts of the system is more effective than for the entire system, and (4) formally describing metamorphic testing is essential.

In recent years, the number of published papers on applying metamorphic testing technology has been growing, and it has been applied in various fields. In bioinformatics, Chen et al. used metamorphic testing for debugging two open-source bioinformatics programs [8]. In the popular field of machine learning, Xie et al. used metamorphic testing to debug classifiers in supervised learning [11]. In 2018, Zhang et al. proposed metamorphic testing for frequent itemset mining problems and used them to verify the

integrity of frequent itemsets [12]. In 2021, Ma et al. used metamorphic testing to verify fake news detection software [21]. Hong et al. used metamorphic testing in erasable-itemset mining in 2022 [13]. Rehman et al. proposed metamorphic testing for machine learning: applicability, challenges, and research opportunities in 2023 [22]. Towey et al. used metamorphic testing on an automated parking system [23]. Chaleshtari et al. proposed metamorphic testing for web system security and used it to make sure the web system is safe [24]. There is still much research in progress.

3 Proposed Metamorphic Relations for High-Utility Itemset Mining

In this section, we describe our proposed metamorphic relations in detail. We first define the notations that we use in the metamorphic relations. We then divide the proposed metamorphic relations into two types: algorithm-independent metamorphic relations, which are used for any utility-mining algorithms, and algorithm-dependent metamorphic relations, which are used for utility-mining algorithms based on two phases.

3.1 Notation

(a) Source and follow-up inputs. A source input refers to an existing test case, and a follow-up input is the result of a source input being converted by a specific input relation [13]. An example of a follow-up input is that the minimum threshold may be set larger than the original (source) one. The symbols of the source and the follow-up inputs are listed in Table 1.

Table 1. Symbols of the source and the follow-up inputs

<i>Source input</i>	<i>Follow-up input</i>	<i>Description</i>
D	D'	The transaction database
UT	UT'	The utility table
r	r'	The minimum threshold

(b) Source and follow-up outputs. A source output refers to the results of a mining program with the given source input, and a follow-up output is the results of the same mining program with the follow-up input [13]. The symbols of the source and the follow-up outputs are listed in Table 2.

Table 2. Symbols of the source and the follow-up outputs

<i>Source output</i>	<i>Follow-up output</i>	<i>Description</i>
$HTWU$	$HTWU'$	The set of high transaction-weighted utilization itemsets
HU	HU'	The set of high-utility itemsets
$ HTWU $	$ HTWU' $	The number of high transaction-weighted utilization itemsets
$ HU $	$ HU' $	The number of high-utility itemsets

3.2 Algorithm-independent Metamorphic Relations

As mentioned above, we divide the metamorphic relations for utility mining into algorithm-independent and algorithm-dependent ones. For algorithm-independent metamorphic relations, the source and the follow-up outputs are high-utility itemsets. Seven algorithm-independent metamorphic relations are proposed and described below.

(a) The first metamorphic relation (MR1): decreasing the minimum threshold. We first consider the variation of the numbers of high utility itemsets when the minimum threshold r is decreased. The first metamorphic relation is defined as follows.

Input relation: $UT' = UT$, $D' = D$, and $r' < r$.

Output relation: $HU' \supseteq HU$ and $|HU'| \geq |HU|$.

In the first metamorphic relation, both the follow-up transaction database D' and utility table UT' are the same as the source transaction database D and the source utility table UT . However, the follow-up minimum threshold r' is less than the source minimum threshold r . In this scenario, if the ratio of the actual utility value of an itemset is larger than the source minimum threshold r , it is certainly larger than the follow-up minimum threshold r' because $r' < r$. Thus, if an itemset is high-utility in the source scenario, it is also high-utility in the follow-up input where $r' < r$. On the contrary, if an itemset is high-utility in the follow-up scenario, it is not necessarily high-utility in the source. Thus, $HU' \supseteq HU$.

We may check the correctness of a utility-mining program according to this metamorphic relation. Since when $HU' \supseteq HU$, $|HU'|$ must be equal to or larger than $|HU|$. $|HU'| \geq |HU|$ may also be regarded as a metamorphic checking rule. The checking of $|HU'| \geq |HU|$ is faster than that of $HU' \supseteq HU$, but the latter has a higher error-finding rate than the former.

(b) The second metamorphic relation (MR2): increasing the maximum threshold. In the first metamorphic relation, the source minimum threshold is larger than the follow-up minimum threshold. On the contrary, we may easily derive the metamorphic relation for the source minimum threshold set less than the follow-up minimum threshold. We thus have the second metamorphic relation when the minimum threshold r is increased as follows.

Input relation: $UT' = UT$, $D' = D$, and $r' > r$.

Output relation: $HU' \subseteq HU$ and $|HU'| \leq |HU|$.

The second metamorphic relation can be easily proved in a way similar to the first one.

(c) The third metamorphic relation (MR3): adding a new item and a new transaction. The third metamorphic relation considers the variation of the numbers of high utility itemsets when we add a new item with its external utility value set at 0 into the source utility table to form the follow-up utility table and append a new transaction including only the new item with an arbitrary quantity to the source database to form the follow-up database. Assume NI is a new item not in the source utility table. We have the third metamorphic relation as follows.

Input relation: $UT' = UT \cup \{(NI, 0)\}$, $D' = D \cup \{(NI, q)\}$, and $r' = r$, where q is an arbitrary positive integer quantity.

Output relation: $HU' = HU$ and $|HU'| = |HU|$.

Note that the inner pair of braces $\{(NI, q)\}$ represents a new transaction with only the item NI and without any other items. Since the follow-up database has a new transaction including only the new item NI with the external utility set at 0, the total utility value of the follow-up database is the same as that of the source database. Because $r' = r$ and the total utility values are the same, the high utility itemsets generated in the source database are also high in the follow-up database. Besides, these original high-utility itemsets cannot be further combined with NI since NI only appears in the new transaction. At last, the new item NI can not be high-utility because its utility value in the follow-up database is 0. Thus, the high utility itemsets in the two databases are completely the same. Thus, $HU' = HU$ and $|HU'| = |HU|$.

Again, $HU' = HU$ is a stricter constraint than $|HU'| = |HU|$ since when $HU' = HU$, $|HU'|$ must be equal to $|HU|$. Choosing the checking of $HU' = HU$ or $|HU'| = |HU|$ is also a trade-off between speed and accuracy.

(d) The fourth metamorphic relation (MR4): appending a new item to old transactions. We next consider the variation of the numbers of high utility itemsets when we add a new item with its external utility set at 0 to the source utility table to form the follow-up utility table and append the item with arbitrary quantities to some transactions in the source database to form the follow-up database. Again, assume NI is a new item not in the source utility table, and SD is an arbitrary subset of a source database including T_1, T_2, \dots, T_k , which are k randomly selected transactions from D . The fourth metamorphic relation is stated as follows.

Input relation: $UT' = UT \cup \{(NI, 0)\}$, $T_i' = T_i \cup \{(NI, q_i)\}$, and $r' = r$, where $T_i \in SD$, $SD \subseteq D$, and q_i is an arbitrary positive integer quantity.

Output relation: $HU' \supseteq HU$ and $|HU'| \geq |HU|$.

Since in the follow-up database, only the new item NI with the external utility of 0 is added to some transactions in the source database, the total utility value of the follow-up database is the same as that of the source database. Because $r' = r$ and the total utility values are the same, the original high utility itemsets generated in the source database are also high in the follow-up database. But these original high-utility itemsets can be combined with NI if they are in the same transactions. The newly generated itemset may appear in fewer transactions than the original one, but the total utility value of the former may still have the chance to be larger than the minimum threshold. Thus, these newly combined itemsets may or may not be high utility depending on their final total utility values. Thus, $HU' \supseteq HU$ and $|HU'| \geq |HU|$.

(e) The fifth metamorphic relation (MR5): Adding a new item and a new transaction (II). Assume TU is the total utility of the source database D . We then consider the variation of the numbers of high utility itemsets when we add a new item with its external utility value set at an arbitrary positive integer smaller than $TU \times r$ to the source utility table to form the follow-up utility table and append a new transaction including only the new item with the quantity of 1 to the source database to form the follow-up database. The fifth metamorphic relation is described as follows.

Input relation: $UT' = UT \cup \{(NI, u)\}$, $D' = D \cup \{(NI, 1)\}$, and $r' = r$, where u is an arbitrary positive integer smaller than $TU \times r$.

Output relation: $HU' \subseteq HU$ and $|HU'| \leq |HU|$.

Because the external utility u of the new item is smaller than $TU \times r$, and the new transaction includes only the new single item with the quantity of 1, the total utility of the new item is u . The total utility of the follow-up dataset thus becomes $TU + u$. Because $r' = r$, the total utility of the new item $< TU \times r < (TU + u) \times r = (TU + u) \times r'$. Thus, the new item cannot be a high-utility itemset in the follow-up database.

Besides, no original high utility itemsets can be combined with the new item since it only appears in the new transaction. Furthermore, because the follow-up minimum threshold value is $(TU + u) \times r'$, which is larger than the minimum threshold utility value $TU \times r$ in the source, some of the source high utility itemsets may be removed because their utility values cannot reach the threshold. Thus, $HU' \subseteq HU$, and certainly $|HU'| \leq |HU|$.

(f) The sixth metamorphic relation (MR6): adding a new item and a new transaction (III). Assume TU is the total utility of the source database D and q is an arbitrary positive integer. The sixth metamorphic relation considers the variation of the numbers of high-utility itemsets when we add a new item with its external utility value set at $\lfloor TU \times r \div q \rfloor$ to the source utility table to form the follow-up utility table and append a new transaction with only the new item with the quantity of $(q - 1)$ to the source database to form the follow-up database. The sixth metamorphic relation is stated below.

Input relation: $UT' = UT \cup \{(NI, \lfloor TU \times r \div q \rfloor)\}$, $D' = D \cup \{(NI, q - 1)\}$, and $r' = r$, where q is an arbitrary positive integer, $q > 1$.

Output relation: $HU' \subseteq HU$ and $|HU'| \leq |HU|$.

In this relation, the total utility of the follow-up dataset becomes $TU + u \times (q - 1)$. The follow-up threshold utility value is $\lfloor TU + u \times (q - 1) \rfloor \times r'$. Because $r' = r$, the follow-up threshold utility value is also $\lfloor TU + u \times (q - 1) \rfloor \times r$. Because u is $\lfloor TU \times r \div q \rfloor$, thus $u \leq TU \times r \div q$. Therefore, $u \times (q - 1) < \lfloor TU + u \times (q - 1) \rfloor \times r$, which can be proven and omitted here. Therefore, the new item may not be in the high utility itemset.

Besides, no original high utility itemsets can be combined with the new item since it only appears in the new transaction. Furthermore, because the follow-up minimum threshold value is $\lfloor TU + u \times (q - 1) \rfloor \times r'$, larger than the source minimum threshold value, some of the source high utility itemsets may be removed. Thus, $HU' \subseteq HU$, and certainly $|HU'| \leq |HU|$.

(g) The seventh metamorphic relation (MR7): adding a new item and a new transaction (IV). The relation considers the variation of the numbers of high-utility itemsets when we add a new item that will become a high-utility itemset in the follow-up database. The seventh metamorphic relation is described as follows.

Input relation: $UT' = UT \cup \{(NI, \lfloor (TU \times r) / ((1 - r) \times q) \rfloor)\}$, $D' = D \cup \{(NI, q)\}$, and $r' = r$, where q is an arbitrary positive integer.

Output relation: $HU' \subseteq HU \cup \{(NI)\}$ and $|HU'| \leq |HU| + 1$.

For this relation, a transaction with a new single item is added to the source database to form the follow-up database. Since no other transactions contain the new item except the new transaction, the total utility of the new item is $q \times \lfloor (TU \times r) / ((1 - r) \times q) \rfloor$, equal to or larger than $q \times (TU \times r) / ((1 - r) \times q)$, which is $TU \times r / (1 - r)$. Besides, the total utility of the follow-up database becomes $TU + q \times \lfloor (TU \times r) / ((1 - r) \times q) \rfloor$,

which is thus equal to or larger than $TU + TU \times r / (1 - r)$, which is $TU / (1 - r)$. Thus, the utility ratio of the new item in the follow-up database is:

$$\begin{aligned}
 & q \times [(TU \times r) / ((1 - r) \times q)] / (TU + q \times [(TU \times r) / ((1 - r) \times q)]) \\
 &= 1 - (TU / (TU + q \times [(TU \times r) / ((1 - r) \times q)])) \\
 &\geq 1 - (TU / (TU / (1 - r))) \\
 &= 1 - (1 - r) \\
 &= r
 \end{aligned}$$

Thus, the new item is a high-utility itemset in the follow-up database. Like the sixth metamorphic relation, no other itemsets can be combined with the new item since it only appears in the new transaction. Because the follow-up minimum threshold value is $TU + q \times [(TU \times r) / ((1 - r) \times q)]$, which is larger than the source minimum threshold value, some of the source high utility itemsets may not be in the follow-up high utility itemsets. Thus, $HU' \subseteq HU \cup \{(NI)\}$ and $|HU'| \leq |HU| + 1$.

3.3 Algorithm-dependent Metamorphic Relations

As mentioned above, the two-phase utility mining algorithm is a popular approach for finding high-utility itemsets. If a program is written based on the two-phase algorithm, some additional metamorphic relations can be designed to check high transaction-weighted utilization itemsets (HTWU), which are intermediate outputs after the first phase. We thus propose seven metamorphic relations (MR8 ~ MR14), which focus on high transaction-weighted utilization itemsets. We omit them here due to the paper length limit.

4 Experimental Evaluation

In this section, we describe our experiments for evaluating the effectiveness of the proposed metamorphic relations. We used the chess_utility database [29], which had 3196 transactions and 75 items, and implemented the two-phase utility mining algorithm in Java to compare algorithm-independent and algorithm-dependent metamorphic relations. To simulate the implementation error in the mining program, we use the μ Java tool, a mutation system for Java programs [16] to modify the codes for generating incorrect programs. Each modified program is called a mutant. The mutation score is defined below:

$$\text{mutation score} = \frac{\text{the number of killed programs}}{\text{the total number of mutant programs}},$$

where a killed program means a mutant program violates an adopted metamorphic relation. If the mutation score of a metamorphic relation is higher, the checking effect of the relation is better. The average mutation scores for the seven proposed algorithm-independent metamorphic relations are shown in Figure 1.

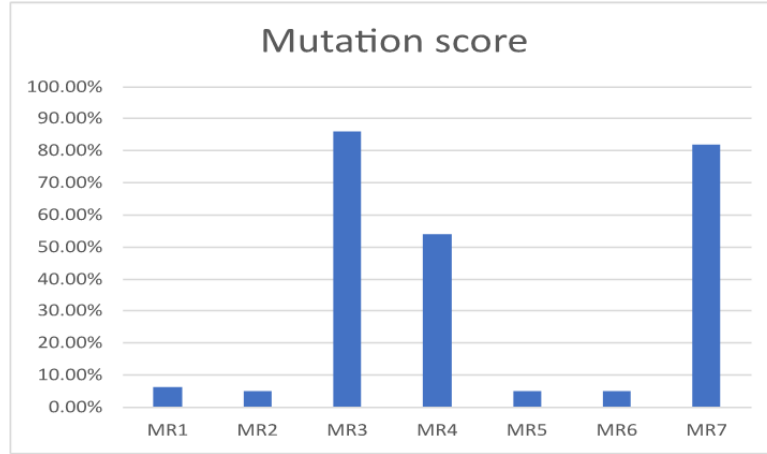


Fig. 1. The mutant scores of the algorithm-independent metamorphic relations

It can be easily seen from the figure that MR3 has the highest mutation score, meaning it has the best capability to detect incorrect programs among the seven metamorphic relations. It is a strict relation because the source and the follow-up outputs must be the same.

The average mutation scores for the seven proposed algorithm-dependent metamorphic relations are shown in Figure 2. Similar to the results in Figure 1, MR10 has the best performance for checking the results after Phase 1 because it has strict conditions.

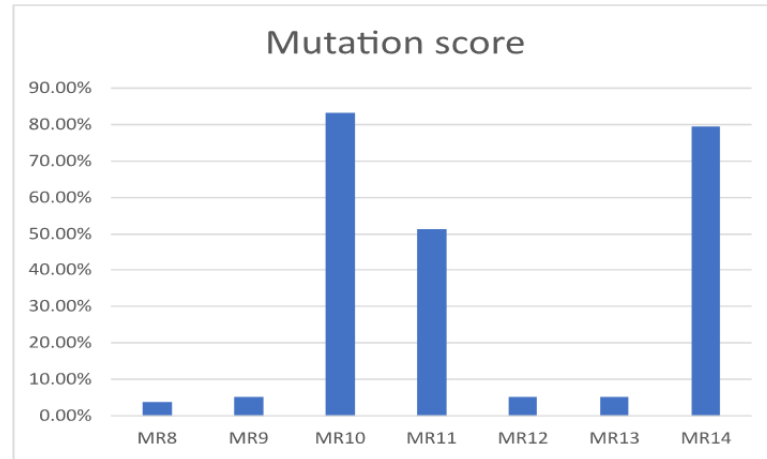


Fig. 2. The mutant scores of the algorithm-dependent metamorphic

We then compare the results within the two groups of metamorphic relations, with the results shown in Figure 3. It can be easily seen that for each pair of corresponding relations, the mutation score of an algorithm-independent metamorphic relation is always higher than its partner algorithm-dependent metamorphic relation. The

phenomenon is reasonable because whenever the mined result after the first phase is wrong, it will still be incorrect after the whole mining process. However, the difference is slight, meaning the algorithm-dependent metamorphic relations can check out most errors in a two-phase utility mining algorithm.

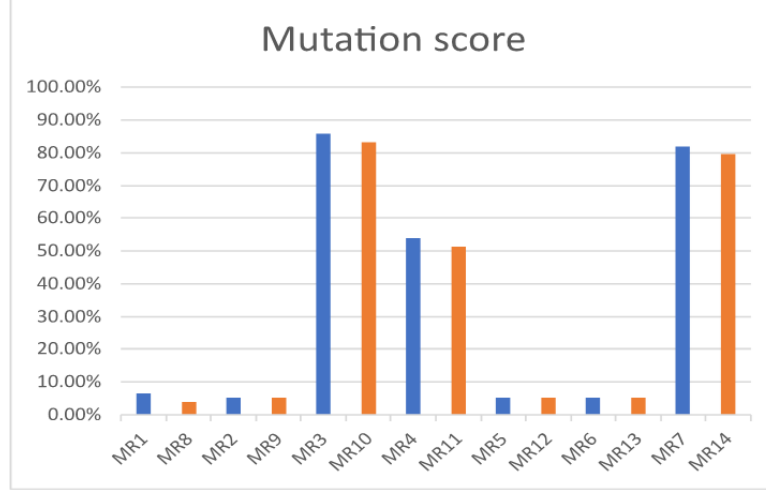


Fig. 3. Comparison of the two groups of metamorphic relations

5 Conclusion and Future Work

Data mining has become important in the present era, which is dominated by vast amounts of data. Most researchers focus on reducing execution time but sometimes overlook the accuracy of the mining programs. Incorrect mining results can lead companies to make misguided decisions. However, the cost of validating these results can be prohibitively high. In order to address this issue, we use metamorphic testing, a lightweight method to verify the accuracy of high-utility itemset mining programs. We have divided the metamorphic testing into algorithm-independent and algorithm-dependent types. We have also designed several metamorphic relations for these two types. For evaluation, We have used the mutation score to validate the performance of these metamorphic relations. In the future, we will attempt to design more metamorphic relations for utility mining and other problems. We will also use other utility-mining algorithms and more datasets for validation.

Acknowledgment

This work was supported by the grant NSTC 109-2221-E390-015-MY3, the National Science and Technology Council, Taiwan.

References

1. R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large database," The ACM SIGMOD International Conference on Management of Data, pp. 207-216, 1993.
2. H. Yao, H. J. Hamilton, and C. J. Butz, "A foundational approach to mining itemset utilities from databases," SDM, pp., 2004.
3. 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, 1998.
4. H. Yao, H. J. Hamilton, and C. J. Butz, "A foundational approach to mining item-set utilities from databases," The Third SIAM International Conference on Data Mining, pp., 2004.
5. H. Yao and H. J. Hamilton, "Mining itemset utilities from transaction database," Data and Knowledge Engineering, Vol. 59, No. 3, pp. 603-626, 2006.
6. Y. Liu, W. Liao, and A. Choudhary, "A two-phase algorithm for fast discovery of high utility itemsets," The Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2005.
7. T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic testing: a new approach for generating next test cases," Tec. rep., Technical Report HKUST-CS98-01, 1998.
8. T. Y. Chen, J. W. Ho, H. Liu, and X. Xie, "An innovative approach for testing bioinformatics programs using metamorphic testing," BMC Bioinformatics, Vol. 10, Issue 1, pp. 24, 2009.
9. S. Segura, G. Fraser, A. B. Sanchez, and A. Ruiz-Cortes, "A survey on metamorphic testing," IEEE Transactions on Software Engineering, Vol. 42, No. 9, 2016.
10. J. W. Zhang, X. Y. Xie, and Z. Y. Zhang, "How reliable is your outsourcing service for data mining? A metamorphic method for verifying the result integrity," International Conference on Software Analysis, Testing, and Evolution, pp. 120-136, 2018.
11. X. Xie, Z. Y. Zhang, T. Y. Chen, Y. Liu, P. L. Poon, and B. Xu, "METTLE: a metamorphic testing approach to assessing and validating unsupervised machine learning systems," IEEE Transactions on Reliability, 2020.
12. J. W. Zhang, X. Y. Xie, and Z. Y. Zhang, "How reliable is your outsourcing service for data mining? A metamorphic method for verifying the result integrity," International Conference on Software Analysis, Testing, and Evolution, pp. 120-136, 2018.
13. T.P. Hong, C.C. Chiu, J.H. Su, and C.H. Chen, "Applicable metamorphic testing for erasable-itemset mining," IEEE Access, Vol. 10, pp. 38545-38554, 2022.
14. T. P. Hong, C. Y. Lin, W. M. Huang, K. S. M. Li, L. S. L. Wang, and J. C. W. Lin, "Using tree structure to mine high temporal fuzzy utility itemsets," IEEE Access, Vol. 8, pp. 153692-153706, 2020.
15. C.-X. Ma and J. Zhang, "DHUI: a new algorithm for mining high utility itemsets," 2009 International Conference on Machine Learning and Cybernetics, pp. 173-177, 2009.

16. Y. S. Ma, J. Offutt and Y. R. Kwon, "MuJava: an automated class mutation system," *Software Testing, Verification and Reliability*, Vol. 15, Issue 2, pp. 97-133, 2015.
17. Pushp and S. Chand, "Mining of high utility itemsets for incremental datasets," 2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), Mauritius, Mauritius, pp. 01-06, 2021.
18. Pushp and S. Chand, "Mining of high utility itemsets with negative utility values for incremental datasets," 2021 International Conference on Computational Performance Evaluation (ComPE), Shillong, India, pp. 431-436, 2021.
19. T. P. Hong, W. T. Hung, W. M. Huang and Y. C. Tsai, "Incremental fuzzy utility mining with tree structure," 2022 IEEE International Conference on Big Data (Big Data), pp. 6202-6206, 2022.
20. J. F. Qu, P. Fournier-Viger, M. Liu, B. Hang and C. Hu, "Mining high utility itemsets using prefix trees and utility vectors," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 10, pp. 10224-10236, 2023.
21. Y. Ma, D. Towey, T. Yueh Chen and Z. Quan Zhou, "Metamorphic testing of fake news detection software," 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC), pp. 1508-1513, 2021.
22. F. U. Rehman and M. Srinivasan, "Metamorphic testing for machine learning: applicability, challenges, and research opportunities," 2023 IEEE International Conference On Artificial Intelligence Testing (AITest), pp. 34-39, 2023.
23. D. Towey et al., "Metamorphic testing of an automated parking system: an experience report," 2023 IEEE 47th Annual Computers, Software, and Applications Conference, pp. 1774-1779, 2023.
24. N. B. Chaleshtari, F. Pastore, A. Goknil and L. C. Briand, "Metamorphic testing for web system security," in *IEEE Transactions on Software Engineering*, vol. 49, no. 6, pp. 3430-3471, 2023.
25. U. Yun et al., "Prelarge-based utility-oriented data analytics for transaction modifications in internet of things," in *IEEE Internet of Things Journal*, vol. 8, no. 24, pp. 17333-17344, 2021.
26. H. Kim et al., "Efficient method for mining high utility occupancy patterns based on indexed list structure," in *IEEE Access*, vol. 11, pp. 43140-43158, 2023.
27. J. C. W. Lin, Y. Djenouri, G. Srivastava and J. M. -T. Wu, "Large-scale closed high-utility itemset mining," 2021 International Conference on Data Mining Workshops (ICDMW), pp. 591-598, 2021.
28. B. Vo et al., "Mining correlated high utility itemsets in one phase," in *IEEE Access*, vol. 8, pp. 90465-90477, 2020.
29. P. Fournier-Viger, C. W. Lin, A. Gomariz, T. Gueniche, A. Soltani, Z. Deng, and H. T. Lam, The SPMF Open-Source Data Mining Library Version 2. Proc. 19th European Conference on Principles of Data Mining and Knowledge Discovery, Part III, Springer LNCS 9853, pp. 36-40.