A Topic Model and Test History-Based Test Case Recommendation Method for Regression Testing

Hirohisa Aman*, Takashi Nakano†, Hideto Ogasawara† and Minoru Kawahara*
*Center for Information Technology, Ehime University, Matsuyama, Ehime 790-8577, Japan
†Corporate Software Engineering & Technology Center, Toshiba Corporation, Kawasaki, Kanagawa 212–8585, Japan

***Abstract*—In the regression testing, an oversight of a regression is a serious problem to be avoided. A test engineer usually selects test cases to rerun for a regression testing. While the selection is a useful expert decision, there is also a risk of missing some important test cases. To support a more effective regression testing, this paper focuses on the following two kinds of data: 1) the similarity of test cases in terms of their topics, and 2) the test history. Then, the paper proposes a hybrid method for recommending test cases in two steps by using the above data. As the thirst step, it recommends test cases which are highly-similar to the manually-selected ones. Then, as the second step, the method recommends the remaining test cases in decreasing order of priority computed by using the test history. The usefulness of the proposed method is proved through an empirical study using a set of regression test data for an industrial product.**

I. Introduction

A software product usually evolves through program modifications which aim at enhancing functionalities or fixing faults of the product, and such a software evolution is required to provide an improved service to the users. However, those modifications may also have risks of creating other faults (regressions) [1]. In order to avoid releasing a faulty version of the product, a careful regression testing—rerunning test cases—should be performed when the product is changed.

To detect all regressions perfectly, it is ideal to rerun all test cases whenever a part of the target product is changed. However, rerunning all test cases would be unrealistic because it requires a lot of time and effort [2] but it may be inefficient to detect regressions: most of all test cases pass almost always, and only a few ones may fail fairly infrequently. Therefore, there have been many studies to prioritize test cases toward a more efficient regression testing [3]. The test case prioritization studies are classified into three types according to what they are mainly based on: (1) the source programs to be tested, (2) the test history and (3) the content of test cases.

The first type of study is to prioritize test cases by analyzing the source programs to be tested. The common way among these studies is to make links between a test case and its corresponding programs which are tested by the test case. When a source program is modified, those studies single out the corresponding test cases which are associated with the modified parts and/or evaluate a test case's degree of closeness to the modified programs (e.g., [4], [5]).

The second type of study utilizes the test history which consists of the testing results such as the version of tested product, the IDs of used test cases and their results (passes or fails). This type of study evaluates a test case's priority by using its failure detection rate in the past and/or its version gap between the current version and the last-run one [6], [7]. While the test history-based methods are not directly related to program modifications, they can be useful to prevent overlooking unexpected regressions which are apparently independent of the modified programs.

The third type of study focuses on the content of test cases. Test cases are either test programs (for an automated test) or test scripts (for a manual test). In this type of study, test cases are analyzed by using the natural language processing techniques such as the morphological analysis and the topic analysis, and their similarities or dissimilarities to each other are evaluated. Then, test cases are prioritized for a better regression testing, by using their similarities/dissimilarities to the set of already selected (being rerun) test cases [8], [9].

When a test engineer performs a regression testing for a product, he/she does not always know anything about the corresponding code modifications made to the product [10]. Furthermore, if the coding work is an (offshore) outsourcing, it would be harder to obtain the details of modifications. In such a case, we cannot utilize the above first type of study which is based on the source program analysis. Since our (the authors') context is in this case, we will focus on the second and the third types of study—using the test history and the content of test cases—in this paper.

For a regression testing, test engineers may select test cases to rerun from their test case pool in accordance with their knowledge and/or experience. We believe those selections are useful because it is a kind of expert decision. However, there would also be a risk of overlooking regressions. Hence, while respecting their first selection, we consider recommending additional test cases which are useful to detect regressions. The key contribution of this paper is to propose a novel method for making a useful recommendation to prevent overlooking regressions by utilizing the test history and the topic analysis to the test cases [11], which is a combination of the above second and third types of study.

The remainder of this paper is organized as follows. Section II explains the test case recommendation in our context and related previous work, then proposes a novel method. Section III reports the empirical study that we conducted to show a usefulness of our proposal. Section IV briefly describes the related work. Finally, Section V presents our conclusion and future work.

II. Effective Test Case Recommendation for Regression Testing

A. Test Case Recommendation Problem

We first explain our context of regression testing. For a regression testing of a product, we have a set of test cases (test case pool). These test cases are written in Japanese and they are test scripts to manually operate the product by a test engineer. At a regression testing, the test engineer knows which functionalities were upgraded and/or which bugs were fixed. Then, the test engineer singles out some test cases from the pool, which seem to be related to the modifications.

Now, in order to reduce a risk of overlooking regressions, we want to recommend adding other test cases which are remained in the pool and yet seem to be useful to detect regressions (see Fig. 1). Since the testing is performed manually, the recommendation set should be minimal because of their testing cost. Our available data are the test history (see Table I) and the content of test cases (see Fig. 2).

Table I shows an example of test history. In the table, symbols "P," "F" and "–" signify the "pass," "fail" and "no-run" of the corresponding test case at the corresponding version, respectively. For example, test case t_2 passed at version v_2 and failed at v_4, and t_3 was not run at v_1. In Table I, v_5 is the version at which we are about to perform a regression testing. Symbol "✓" means that the corresponding test cases are selected by the test engineer; the remaining test cases shown as "?" are the candidates of our recommendation.

Our goal is to recommend useful test cases from the candidate set. A better recommendation means that more regressions are detected by fewer test cases, i.e., test cases with higher priorities are more likely to detect regressions. This is formally defined as the following test case recommendation problem.

Definition 1 (The test case recommendation problem):

Given the set of test cases, T, which consists of n test cases: $T = \{t_i\}_{i=1}^{n}$. Suppose a test engineer selected m test cases $t_{1,j}$ from T (for $j = 1, \ldots, m$); Let $H = \{t_{1,j} | t_{1,j} \in T\}_{j=1}^{m}$. Then, recommend the remaining test cases in decreasing order of priority, as a sequence $(t_{2,r})_{r=1,\ldots,n-m}$ where $(t_{2,r} \in T \bigwedge t_{2,r} \notin H)$ and $t_{2,r}$ has a higher priority than $t_{2,r'}$ if $r < r'$ (for $r = 1, \ldots, n-m$).

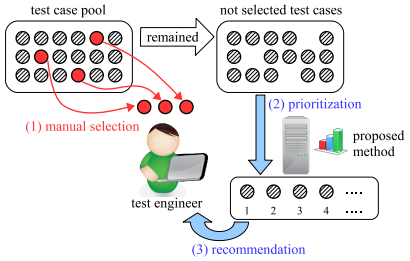


Fig. 1. Outline of test case recommendation.

Hover the mouse cursor over the "?" icon. Then, check if the corresponding help message is displayed in a pop-up window.

Fig. 2. Example of test case (translated into English).

TABLE I
Example of test history.

test case	version v_1	v_2	v_3	v_4	v_5
t_1	P	–	–	–	?
t_2	F	P	P	F	✓
t_3	–	P	–	–	?
t_4	–	F	P	P	?
t_5	P	–	–	–	✓
⋮	⋮	⋮	⋮	⋮	⋮

(P: pass; F: fail; –: no run; ✓: selected; ?: candidate)

The usefulness of recommendation is evaluated by the area under the curve of the Alberg diagram (AUCA)[1] [12]. A higher AUCA value means a better recommendation because more test cases succeeded in detecting regressions, in an earlier phase of the regression testing.

□

Figure 3 shows examples of Alberg diagrams. The best case is in Fig. 3 (a) since all regressions are detected by the higher priority test cases. If some high-priority test cases fail to detect regressions, the diagram becomes like Fig. 3 (b) and its AUCA value (the area under the curve) gets smaller than (a).

B. Test Case Recomendations Using Test History and/or Content of Test Cases

To recommend (prioritize) test cases, we use the test history and the content of test cases. In this subsection, we describe methods proposed in the previous work.

In the test history-based prioritization, the following two types of data are mainly focused:

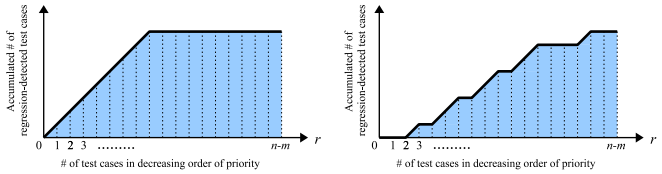
[Type-1] Whether a test case was run or not at a version.

[Type-2] Whether a test case failed or not at a version.

1) Application of the Exponential Weighted Moving Average and the Exponential Smoothing: Kim and Porter [6] proposed to leverage the exponential weighted moving average and the exponential smoothing [13]. For a pair of test case t_i and version v_j, they defined the selection probability of t_i as:

$$P(t_i, v_j) = \alpha \cdot h_{ij} + (1 - \alpha) P(t_i, v_{j-1}) \ , \qquad (1)$$

where $h_{ij} \in \{0, 1\}$ is the test history observation for t_i at v_j, and α is a smoothing constant used to weight individual history observations ($0 \leq \alpha \leq 1$); $P(t_i, v_0) = h_{i1}$.



(a) the best case (b) another case

Fig. 3. Examples of Alberg diagrams.

[1]The vertical axis signifies the accumulated number of faults (regressions) in the original Alberg diagram. However, we replace it by the accumulated number of regression-detected test cases because we cannot trace the faults in the source code; While the test cases and the test history are available, the source code is not available in our context.

A higher $P(t_i, v_j)$ means a higher priority of t_i for a regression testing at the next version, v_{j+1}. Kim and Porter proposed to use Eq. (1) for both the above two types of data.

For type-1 data, let $h_{ij} = 1$ if t_i was "not run" at v_j; otherwise, $h_{ij} = 0$. Then, they used Eq. (1) with a low α, e.g., $\alpha = 0.1$. If t_i has not been run for more versions until v_j, $P(t_i, v_j)$ gets larger. For example, let us consider t_1 shown in Table I. Here, we have $(h_{1k}) = (0, 1, 1, 1)$. When $\alpha = 0.1$, their selection probabilities are upgraded from v_1 to v_4 as: $P(t_1, v_k) = 0 \ \rightarrow \ 0.1 \ \rightarrow \ 0.19 \ \rightarrow \ 0.271$.

For type-2 data, let $h_{ij} = 1$ if t_i "failed" at v_j; otherwise, $h_{ij} = 0$. In this case, they proposed to use Eq. (1) with a high α, e.g., $\alpha = 0.9$. If t_i has consecutively failed for more versions until v_j, $P(t_i, v_j)$ gets larger. Let's take t_2 shown in Table I for example. We have $(h_{2k}) = (1, 0, 0, 1)$. When $\alpha = 0.9$, their selection probabilities change from v_1 to v_4 as: $P(t_2, v_k) = 0.9 \ \rightarrow \ 0.09 \ \rightarrow \ 0.009 \ \rightarrow \ 0.9009$.

For the sake of convenience, we will call the above methods using the selection probabilities for type-1 and type-2 as "$SP1$" and "$SP2$," respectively, in the following sections.

2) Application of the Mahalanobis-Taguchi Method: Aman et al. [7] proposed to integrate the above two types of data using the notion of the Mahalanobis-Taguchi method[2] [14].

In [7], Type-1 and type-2 data are measured by the following two metrics GLC and FR, respectively.

1) The Gap between the Last run version and the Current version (GLC): For a test case t_i, GLC(t_i) is the number of consecutive versions at which t_i was not run until the current version.
2) The Failure Rate (FR): For a test case t_i, FR(t_i) is the failure rate of t_i until the current version. That is to say, it is the number of versions failed by t_i, divided by the number of versions at which t_i was run.

For example, GLC(t_1) = 4, GLC(t_4) = 1, FR(t_1) = 0 and FR(t_2) = $2/4$ in Table I, where the current version is v_5.

A test case having a greater GLC value has not been run for more versions after its last run. Therefore, such a test case may have a higher risk of overlooking regressions. While test engineers often select test cases which seem to be related with the functionalities upgraded at the version, unexpected parts might include regressions. It is better to rerun other test cases as well, even if they seem to have no relation with the changes.

A test case having a higher FR value has a better track record for finding a regression in the past. Such a test case may test a part which is likely to be faulty in the product, and we can expect a higher ability to find a regression again. It is better to rerun as many such test cases as possible.

On both of these metrics, value 0 represents the corresponding test case has the lowest expectation to find a regression. Thus, the distance between $\boldsymbol{x_i}^T = (\mathrm{GLC}(t_i), \mathrm{FR}(t_i))$ and $\mathbf{0}^T = (0, 0)$ can express the worth of t_i to rerun in terms of these metrics. A straightforward way of computing the distance is $d_E(\boldsymbol{x_i}) = \sqrt{(\boldsymbol{x_i} - \mathbf{0})^T(\boldsymbol{x_i} - \mathbf{0})}$, which is the Euclidean distance. However, it has a lack of consideration for the dispersion of data. Since metric GLC has a wider range than FR, GLC can have a greater impact on the computed distance. The following Mahalanobis distance is a better choice to consistently integrate the effects of two metrics:

$$d_M(\boldsymbol{x_i}) = \sqrt{(\boldsymbol{x_i} - \mathbf{0})^T S^{-1} (\boldsymbol{x_i} - \mathbf{0})} \ , \qquad (2)$$

where S is the covariance matrix [15] of GLC values and FR values, and S^{-1} is its inverse matrix. Since it uses the covariance matrix, the Mahalanobis distance can consider not only the dispersion on each axis (a certain metric) but also their covariances of data, i.e., correlations between metrics.

For example, if we have the following covariance matrix S,

$$S = \begin{bmatrix} 4.000 & -0.200 \\ -0.200 & 0.050 \end{bmatrix}, \quad S^{-1} = \begin{bmatrix} 0.3125 & 1.2500 \\ 1.2500 & 25.000 \end{bmatrix},$$

we obtain $d_M(\boldsymbol{x_1}) \simeq 2.24$, $d_M(\boldsymbol{x_3}) \simeq 1.68$ and $d_M(\boldsymbol{x_4}) \simeq 1.98$. On the other hand, by using the Euclidean distance, we get $d_E(\boldsymbol{x_1}) = 4$, $d_E(\boldsymbol{x_3}) = 3$ and $d_E(\boldsymbol{x_4}) \simeq 1.05$. That is to say, although t_1 is the worthiest in $\{t_1, t_3, t_4\}$ for both the Euclidean distance and the Mahalanobis one, the second worthiest test cases are different between two distances; In the Euclidean distance, the value of GLC seems to be dominant.

For the sake of convenience, we hereinafter call the above method using the Mahalanobis distance as "MD."

3) Content-Based Clustering of Test Cases and Combination with MD: We have focused on the content of test cases in the past [9]. In [9], we proposed to recommend test cases in the following two steps:

1) First, categorize the test cases into some clusters in terms of their similarities, based on their mutual words.
2) Then, recommend test cases from the same cluster as the manually selected test cases[3]. The recommendation is made by the MD method mentioned above.

When an engineer selected a test case t_x and there is similar but not identical another test case t_y, the engineer maybe should run not only t_x but also t_y. For example, suppose we want to test two different functions, A and B, under a certain condition: t_x executes in the order (A,B), and t_y does in the opposite order (B,A). In such a case, the contents of t_x and t_y must be similar. However, since they are not identical, we should not skip over either t_x or t_y; we have to rerun both of them. Some test engineers might rerun only t_x, and overlook a regression which would have been detected by t_y.

While our previous method [9] partially succeeded to recommend test cases which can detect regressions, it had the following challenge: how to deal with the variation of words among test cases. This is from a linguistic feature in our dataset. Our dataset was the set of test cases written in a natural language, Japanese. Hence, different people may write different phrases with using different words for the same or similar purpose. Especially, Japanese tends to have a high

[2]The Mahalanobis-Taguchi method is one of statistical quality control methods, which leverages the Mahalanobis distance to detect abnormal (poor quality) objects, etc. See [14] for the details.

[3]In the case of Table I, the manually selected test cases are t_2 and t_5.

vagueness of phrase. Since the previous work considered the similarity between test cases by using the exact match of mutual words, such a vagueness may be a serious threat to validity. To solve the above challenge in our previous work, we will propose to apply the topic model in the next subsection.

C. Proposal

The topic model is a popular model for analyzing documents written in a natural language [16], [17]. In the modeling, the words are extracted from the documents and their co-occurrence relationships are analyzed. Through the co-occurrence analysis, each word is liked to one or more topics, where topics are latent variables (see Fig. 4). This liking work is like a clustering of words, but it does not limit word's cluster to a specific one. Finally, each document is associated to one or more topics. While topics do not explicitly appear in a document, the document's feature is expressed by the feature vector $(p_1, \ldots, p_k)$ where p_i denotes the probability that the document has the i-th topic; k is the number of topics. The topic modeling is performed by the latent Dirichlet allocation (LDA) [18] which is a popular statistical method to estimate the latent topics in documents. There are implementations of the LDA for natural language documents: for example, `topicmodels` package[4] is available for `R`[5].

By using the topic model, we can solve the challenge mentioned in Section II-B3, i.e., vagueness of words. We now can consider the similarity between test cases by not the mutual words but the latent semantics of words. That is to say, since the feature vector $(p_1, \ldots, p_k)$ presents the membership degrees to the topics, the semantic similarity between test cases can be computed by using the feature vectors. We define the similarity between test cases t_x and t_y as

$$sim(t_x, t_y) = 1 - \frac{1}{k} \sum_{i=1}^{k} |p_{xi} - p_{yi}| \ , \qquad (3)$$

where k is the number of topics, and p_{xi} and p_{yi} are the probabilities that t_x and t_y have the i-th topic, respectively. It is based on the Manhattan distance, and the reason why we use it is from the experimental results by Thomas et al. [8].

Now we propose a test case recommendation method as the following two steps:

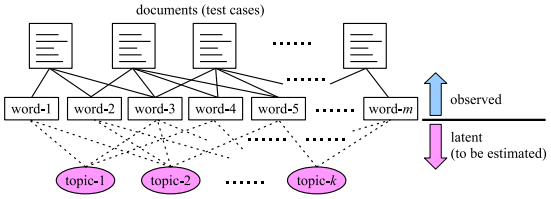


Fig. 4. Image of latent topics.

[4]https://cran.r-project.org/web/packages/topicmodels/

[5]`R` is one of the popular free software environments for statistical computing and graphics. https://www.r-project.org/

[Step-1] Suppose m test cases $\{ t_{1,i} \}_{i=1,\ldots,m}$ are manually selected by a test engineer. Define the similarity of a remaining test case t_x to the selected set as

$$s(t_x) = \min_{i=1,\ldots,m} [\ sim(t_x, t_{1,i}) \ ] \ . \qquad (4)$$

Then, recommend test cases having high similarities with the manually-selected set.

[Step-2] Recommend the remaining test cases by the $SP1$, $SP2$ or MD method.

The first step uses only the similarity of test case. If a highly-similar test case is remained (not selected), it may cause an overlooking of regression. We will decide an appropriate threshold of similarity in our empirical study (Section III). In the second step, we follow the conventional prioritization method ($SP1$, $SP2$ or MD). The above proposal is the key contribution of this paper, which combines the previous methods mentioned in Section II-B: Once a manually-selected "seed" subset of test cases is given, the proposed method automatically prioritize the remaining test cases in the pool.

III. Empirical Study

A. Dataset

To examine our proposal, we obtained data of regression testing for a Web-based system which has been developed and maintained by Toshiba corporation[6]. We have 300 test cases ($n = 300$) and a test history for 13 versions $\{v_j\}_{j=1}^{13}$ of the product. For this study, we manually reran all of 300 test cases at the latest version. It took 539 minutes and revealed that there are regressions overlooked by a past regression testing—22 test cases failed at the latest version. The oversight was occurred at version v_7. For v_7, seven ($m = 7$) test cases were manually selected from the test case pool, but more test cases should be selected to detect regressions earlier.

In this section, we examine if the proposed method makes a better recommendation for the regression testing at v_7, where we consider v_7 to be the current version for computing metric values mentioned above.

B. Procedure

Our empirical study is conducted in the following steps.

1) Build a topic model of our test cases. Prior to the topic modeling, we have to extract words from the test cases. Since they are written in Japanese, we perform a tokenization, a stemming, an elimination of stop words, a morphological analysis, and an extraction of words[7] by `MeCab`[8] which is a popular analyzer for Japanese.
   For a model construction, the number of topics (k) must be given as a key parameter.
2) Given manually-selected test cases $\{t_{1,j}\}_{j=1,\ldots,m}$. For each test case t_x remained in the test case pool, compute $s(t_x)$ according to Eq.(4).

[6]We omit the details of the product since they are confidential data.

[7]Nouns, verbs and adjectives are extracted in this study.

[8]http://taku910.github.io/mecab/

3) Recommend highly-similar test cases $\{t_{2,r}|s(t_{2,r}) \geq \tau\}$ in decreasing order of similarity, where τ is the threshold of similarity; we will empirically decide τ.
4) Recommend the remaining test cases by one of the conventional history-based methods—$SP1$, $SP2$ or MD.

C. Results

To decide the number of topics (k), we first built a topic model with many ($= 100$) topics and performed the principal component analysis[9]. Because the top 19 principal components can explain over 90% information of the original space, we decided the number of topics to be 19 in this empirical study.

We also have to decide the threshold of similarity (τ). Since we have no theoretical basis to decide a proper τ, we will try some values: τ_x (for $x = 1, 2, 3, 4, 5$) where τ_x corresponds to the top x% in the similarity distribution.

Table II shows the AUCA values of recommendations. In the table, the row labeled as "history only" presents the AUCA values of the conventional history-based recommendations: $SP1$ ($\alpha = 0.1$), $SP2$ ($\alpha = 0.9$) and MD. Each row of τ_x (for $x = 1, \ldots, 5$) gives the AUCA values of recommendations made by the proposed method using $\tau = \tau_x$ and the corresponding column's history-based method ($SP1$, $SP2$ or MD). A percentage shown in parentheses denotes the difference of the corresponding AUCA values between the proposed method and the conventional (history only) one: for example, the AUCA value of the recommendation made by using only $SP2$ is 4252, and the AUCA value of the one made by the proposed method using τ_1 and $SP2$ is 4339, so the percentage of difference is obtained as: $(4339 - 4252)/4252 \simeq 2.0\%$.

For each column in Table II, the highest AUCA value is indicated by boldface. Regardless of the used history-based method, the proposed method with $\tau = \tau_1$ or τ_2 show the best performance in recommending test cases in our dataset.

D. Discussions

Our empirical results showed that the proposed method using $\tau = \tau_1$ or τ_2 is the best, i.e., it is useful to rerun the most 1% or 2% similar test cases at first.

TABLE II
EMPIRICAL RESULTS (AUCA VALUES)

	history-based method		
	$SP1$	$SP2$	MD
history only	5166	4252	5318
τ_1	**5409** (+4.7%)	4339 (+2.0%)	**5386** (+1.3%)
τ_2	5374 (+4.0%)	**4390** (+3.2%)	5380 (+1.2%)
τ_3	5320 (+3.0%)	4333 (+1.9%)	5325 (+0.1%)
τ_4	5265 (+1.9%)	4295 (+1.0%)	5269 (−0.9%)
τ_5	5211 (+0.9%)	4238 (−0.3%)	5212 (−2.0%)

[9]To decide the number of topics, we first tried using `ldatuning` package of `R`, https://cran.r-project.org/web/packages/ldatuning/vignettes/topics.html, which provides topic evaluations using four metrics. Unfortunately, these four metrics showed different tendencies and it was hard to decide a specific number of topics. Therefore, we dare to use the principal component analysis as a substitute way of determining the number of topics while it is not orthodox. We need a further study for a better way in the future.

Because of their high similarities, the test engineer might omit them as unimportant ones: Suppose there is a highly-similar pair of test cases, t_x and t_y. When the engineer reran t_x and it passed, he/she would highly expect that t_y also passes since the highly-similar t_x passed. Then, he/she might consider that it does not matter to omit a rerunning of t_y. However, the AUCA value gets higher by prioritizing highly-similar test cases first, so the results prove an importance of focusing on similar test cases. Figure 5 (a) presents the Alberg diagrams of the conventional method and the proposed one using τ_1 with MD (whose X-axis is limited to 100). At the kickoff point in Fig. 5 (a), highly-similar test cases successfully detect regressions which were overlooked by manually-selected ones.

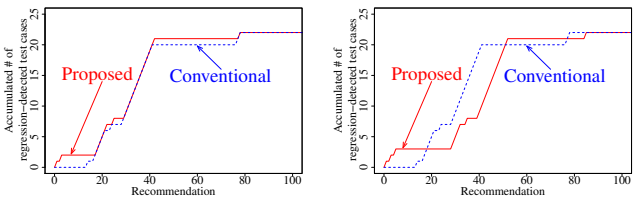
On the other hand, the usefulness of recommending similar test cases seems to be limited to highly-similar ones. Indeed, τ_5 showed the lowest AUCA values for all methods (see Table II). Figure 5 (b) presents the Alberg diagrams of the conventional method and the proposed one using τ_5 with MD. Although highly-similar test cases are really useful (see Fig. 5(a)), recommending more test cases seems to delay the growth of regression detections (see Fig. 5(b)). Since a regression testing with only similar test cases may have a lower coverage, such a delay of growth in detecting regressions would be caused.

E. Threats to Validity

While our recommendation of highly-similar test cases worked for a successful detection of regressions in this empirical study, it might depend on the product's features, the way of testing, or even who did the regression test. To prove a generality of our results, we also need to perform further experiments using other products in the future.

Since we could not trace a link between a code change and a test case in our context, we have discussed the efficiency of testing with the number of "failed test cases" but not the number of "faults." That is to say, two or more test cases might detect the same fault. This is our big threat to validity and limitation in this study. In order to analyze the impact of such a difference of viewpoint, we have to do further analyses using more detailed testing and fault data in the future.

We used the LDA for our topic modeling. While the LDA is the most popular method, the performance depends on the parameters used for the computation. We adopted the default parameters of the `lda` function included in `topicmodels` package. Although the default parameters might not be appropriate to the test cases written in Japanese, we did not tune the



(a) $\tau = \tau_1$ (b) $\tau = \tau_5$

Fig. 5. Comparisons of Alberg diagrams.

parameters because the tuning can be another threat to validity. A further analysis on the parameter tuning is our future work.

IV. Related Work

Thomas et al. [8] utilized the topic model to prioritize test cases. In their method, dissimilarities between test cases (programs) are computed by using the Manhattan distance of corresponding topic vectors. They proposed to iteratively select test cases in decreasing order of dissimilarity to the already-selected test cases in order to expand the test coverage. Hemmati et al. [19] applied the notion of the topic-based prioritization proposed in [8] to test cases written in a natural language: they prioritized test cases by maximizing dissimilarities to already-prioritized test cases in terms of their topics. Our fundamental way of evaluating the similarity/dissimilarity between test cases is common to [8], [19]. While these previous work prioritized test cases to get a higher diversity with a fewer test cases, we have another viewpoint—we dare to recommend test cases which are highly-similar to manually-selected ones. The aim of such an opposite way of prioritization is to avoid a human error. Since such our prioritization would produce a lower coverage, we have a hybrid strategy: after we recommend a few highly-similar test cases, we switch our strategy to the history-based method.

Saha et al. [20] proposed to reduce a test case prioritization problem to an information retrieval one. In their approach, a code change is corresponding to a query, and test cases are regarded as a set of documents. Once a code change is given, test cases are prioritized in terms of the similarity to the change, where the similarity is computed by the tf/idf term weighted vector space model [21]. They reported that the proposed method works better to effectively prioritize test cases through an empirical study using open source projects, and the usefulness of leveraging a natural language processing-based similarity evaluation for prioritizing test cases is proved. Although we cannot get the code change information in our context, it is common that test cases are prioritized based on the similarity to a seed data—while Saha et al. focused on a code change, we used a manually-selected test cases.

V. Conclusion

To enhance the manual regression testing, we focused on the similarity between test cases using the topic model, and proposed a novel method for recommending test cases. When some test cases are manually selected for a regression testing, the proposed method recommends additional test cases in the following two steps. As the first step, the method recommends test cases which are highly-similar to the manually-selected ones. Then, as the second step, the method recommends test cases in decreasing order of priorities which are computed by the conventional test history-based method. Through an empirical study, we proved that the proposed method successfully enhances the performance of regression testing. Notice that the first-step recommendation is limited to "highly" similar test cases only in order to prevent a human error in the manual selection; if we simply prioritized all test cases in decreasing order of similarity, it would not effectively work to detect various regressions. The proposed method uses both the test case similarity and the test history in a hybrid manner.

Our future work includes: 1) a further study to automatically decide a better threshold of the similarity between test cases, and 2) a further analysis using other test data of other products to examine the general usefulness of the proposed method.

Acknowledgment

The authors would like to thank the anonymous reviewers for their helpful comments on an earlier version of this paper.

References

[1] C. Jones, *Applied Software Measurement: Global Analysis of Productivity and Quality*, 3rd ed. New York, McGraw-Hill, 2008.

[2] J. Hartmann and D. J. Robson, "Techniques for selective revalidation," *IEEE Software*, vol. 7, no. 1, pp. 31–36, Jan. 1990.

[3] S. Yoo and M. Harman, "Regression testing minimisation, selection and prioritisation: A survey," *Softw. Testing, Verification & Rel.*, vol. 22, no. 2, pp. 67–120, Mar. 2010.

[4] D. Jeffrey and N. Gupta, "Test case prioritization using relevant slices," in *Proc. 30th Annual Int'l Comp. Softw. & App. Conf.*, vol. 1, Sept. 2006, pp. 411–420.

[5] S. Mirarab and L. Tahvildari, "A prioritization approach for software test cases based on bayesian networks," in *Proc. 10th Int'l Conf. Fundamental Approaches to Softw. Eng.*, Mar. 2007, pp. 276–290.

[6] J.-M. Kim and A. Porter, "A history-based test prioritization technique for regression testing in resource constrained environments," in *Proc. 24th Int'l Conf. Softw. Eng.*, May 2002, pp. 119–129.

[7] H. Aman, Y. Tanaka, T. Nakano, H. Ogasawara, and M. Kawahara, "Application of Mahalanobis-Taguchi method and 0-1 programming method to cost-effective regression testing," in *Proc. 42nd Euromicro Conf. Softw. Eng. & Advanced App.*, Aug. 2016, pp. 240–244.

[8] S. W. Thomas, H. Hemmati, A. E. Hassan, and D. Blostein, "Static test case prioritization using topic models," *Empir. Softw. Eng.*, vol. 19, no. 1, pp. 182–212, Feb. 2014.

[9] H. Aman, T. Nakano, H. Ogasawara, and M. Kawahara, "A test case recommendation method based on morphological analysis, clustering and the Mahalanobis-Taguchi method," in *Proc. 10th IEEE Int'l Conf. Softw. Testing, V. & V. Workshops*, Mar. 2017, pp. 29–35.

[10] M. J. Arafeen and H. Do, "Test case prioritization using requirements-based clustering," in *Proc. 6th Int'l Conf. Softw. Testing, V. & V.*, Mar. 2013, pp. 312–321.

[11] T. K. Landauer, D. S. McNamara, S. Dennis and W. Kintsch, Ed., *Handbook of Latent Semantic Analysis*. London, Routledge, 2014.

[12] N. Ohlsson and H. Alberg, "Predicting fault-prone software modules in telephone switches," *IEEE Trans. Softw. Eng.*, vol. 22, no. 12, pp. 886–894, Dec. 1996.

[13] R. G. Brown, *Statistical Forecasting for Inventory Control*. New York, McGraw-Hill, 1959.

[14] G. Taguchi, S. Chowdhury, and Y. Wu, *The Mahalanobis-Taguchi System*. New York, McGraw-Hill, 2001.

[15] G. J. G. Upton and I. Cook, *A Dictionary of Statistics*, 2nd ed. Oxford University Press, 2008.

[16] T. Hofmann, "Probabilistic latent semantic indexing," in *Proc. the 22nd Int'l ACM SIGIR Conf. Research & Dev. in IR*, Aug. 1999, pp. 50–57.

[17] D. M. Blei, "Probabilistic topic models," *Commun. ACM*, vol. 55, no. 4, pp. 77–84, Apr. 2012.

[18] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Machine Learning Research*, vol. 3, pp. 993–1022, Jan. 2003.

[19] H. Hemmati, Z. Fang, M. V. Mäntylä, and B. Adams, "Prioritizing manual test cases in rapid release environments," *Softw. Testing, Verification & Rel.*, vol. 27, no. 6, pp. e1609:1–25, Sept. 2017.

[20] R. K. Saha, L. Zhang, S. Khurshid, and D. E. Perry, "An information retrieval approach for regression test prioritization based on program changes," in *Proc. 37th Int'l Conf. Softw. Eng.*, May 2015, pp. 268–279.

[21] J. Zhou, H. Zhang, and D. Lo, "Where should the bugs be fixed?–more accurate information retrieval-based bug localization based on bug reports," in *Proc. 34th Int'l Conf. Softw. Eng.*, June 2012, pp. 14–24.