Investigation of Opportunities for Test Case Selection Optimisation Based on Similarity Computation and Search-Based Minimisation Algorithms

Eike Steffen Reetz[1,2], Daniel Kuemper[1], Klaus Moessner[2], Ralf Tönjes[1]

[1]Faculty of Engineering and Computer Science, University of Applied Sciences Osnabrück
Osnabrück, Germany
Email: {e.reetz, d.kuemper, r.toenjes} @hs-osnabrueck.de
[2]Centre for Communication Systems Research, University of Surrey
Guildford, United Kingdom
Email: k.moessner@surrey.ac.uk

***Abstract*—Test Case Diversity investigations promise to reduce the number of Test Cases to be executed whereby addressing one of the drawbacks of automated model-based testing. Based on the assumption that more diverse Test Cases have a higher probability to fail, algorithms for distance analysis and search based minimisation techniques can help to enhance the quality of selection. This work discusses the application of Hamming Distance and Levenshtein Distance to compute similarity scores and outlines how Random Search and Hill Climbing can be applied to the problem of group optimisation based on pairwise Test Case similarity scores. The evaluation results, conducted with a test framework for automated test derivation and execution for IoT-based services, indicates that proposed Group Hill Climbing algorithm can outperform Random Search and at the same time utilising less computation time. The inclusion of the sequence-based Levenshtein algorithm shows advantages over the utilisation of the set-based Hamming-inspired scoring methodology.**

Keywords–Model-based testing; Test Case Diversity, Data Analysis, Hill Climbing, Optimisation Problem.

I. INTRODUCTION

Model-based testing offers the ability for (semi-) automated Test Case (TC) creation and execution based on machine interpretable software specifications. Concepts to derive TCs out of a test model result in a lot of TCs, which can not be executed within a reasonable time without expensive testing costs [1]. Therefore, it is eminent to identify which TC and which test data should be selected for execution to ensure the best target test coverage within given time and resources. The underlying assumption is that automated test case creation can guarantee a wide test target coverage only by creating partly redundant TCs. As a consequence, the process of selecting test data and TCs, which have the highest possibility to identify failures, is crucial for a successful appliance of model-based testing paradigms for application testing.

The present work follows a similarity investigation approach, which tries to identify the most diverse TCs for test execution based on a pairwise similarity between all TCs. This approach can improve the selection of TCs, if parts of the TCs have redundancies and the removal of these redundant TCs have a lower impact to the fault detection rate than randomly removing TCs. The present work tries to enhance the understanding of the application of distance algorithm to the problem of TC reduction whereby investigating the impact off set and sequence-based distance algorithm. The pairwise calculation of the distances of different TCs results in a NP-hard problem [2] if it comes to the selection of a group of TCs out of all TCs (set cover problem). One of the open questions is, if specific search-techniques can enhance the performance compared to random search with comparable computation time effort. This can only be the case if the distance between TCs are not normally distributed and correlations exist for example for TC neighbours. Within this work concepts of the identification of local optimums as introduced by Hill Climbing are applied to the problem of group optimisation of the TC selection.

The evaluation of the work is based on a finite state machine, which is utilised to create TCs based on the W-Method [3]. The results of the experiments indicate that a proposed Group Hill Climbing algorithm perform better than random search with 100 repetitions and on the same time requires less computation time. Positive impact of Group Hill Climbing is identified in case of the utilisation of Levenshtein, which provides a higher granularity between the computed similarity scores.

The paper is organised as follows. After the related work presented in Section II, the investigated algorithm of distance calculation and group optimisation based on search methodology and their application of the TC problem are described in Section III. Afterwards, the evaluation setup and the example test model is presented and the test process is outlined in Subsection IV-A. In Section IV the evaluation findings are highlighted and the conclusion wraps up the paper in Section V.

II. RELATED WORK

Related concepts to reduce the number of TCs exist based on i) test execution history, ii) test purposes and iii) similarity investigations (other approaches especially for white-box testing exists cf. [4]). TC reduction based on history is one oft the widely used techniques for regression tests [5]. The knowledge which TCs failed in previous versions of the System Under Test (SUT) is utilised to select the TCs that are executed during the current test procedure. Other works extended this approach by categorising SUTs and assuming that there is a correlation between failures that occur at different SUT of the same category [6]. Nevertheless, for this approach it is curial to have a good history base. The assumption of correlation between failures in different versions or SUTs of the same categories has not been proven so far as been valid for all kinds of software (with our best knowledge). One methodology to

 ISBN: 978-1-61208-370-4

overcome a missing execution history is based on mutation testing. For example, Zhang et. al. [7] combined mutation testing with test reduction and TC selection/prioritisation. A more straightforward technique is the reduction of the target coverage [8]. By limiting the area of interest it is possible to reduce the number of TCs, but the question remains, which parts are more important than others. The MINTS tool combines coverage, history and costs data for the test case selection [9]. It models the multi-dimensional minimisation problem as a binary Integer Linear Programming (ILP) and can utilise different ILP solvers. The results indicate that this approach can be as good as classical heuristic techniques. Although, the approach is only applicable for regression testing and white-box testing. In advance, it remains unclear how to proceed if no optimal solution exists (e.g contradicting requirements). Other approaches also try to use scenarios or TC weighting mechanisms by including the knowledge of experts [10].

Recent trends in software engineering indicate that search-based optimisation techniques are a promising candidate for several software issues such as requirements, project planning, testing and re-engineering optimisations [11]. Approaches for test case selection (often called prioritisation within this context) include the utilisation of greedy [12] and clustering [13][14] algorithms for white-box testing. The results indicate that cluster-based algorithm can outperform traditional coverage-based TC selection techniques by including human input. Different to these approaches the outlined work follows a model-based testing approach without human interaction within the TC selection process. One of the first researchers who used these search-based approaches for model-based testing was Cartaxo et. al. [15], who tried to select the less similar TCs while maximising the state or transition coverage of the test model with the remaining TCs. Hemmati et. al [16] extended this approach by utilising several similarity functions and minimisation algorithm and applied them to two larger SUTs. The findings indicate that the choice of technique significantly influences the performance. Although, it remains open if the results can be transferred to other SUT and if there is a general setup which is always the best. While the previous results from Cartaxo and Hemmati focus on maximisation of the fault detection rate with the minimum number of TCs, the outlined work focus on validating the average group similarity of the selected TCs. Different to previous approaches the present work focuses on a more precise interpretation of the methodology and tries to make the results better repeatable and comparable to enable a generalisation of results.

III. ALGORITHMS

For the followed TC diversity approach it is necessary to identify the similarity between TCs. Therefore, the starting influence factor is, which information is included into the comparison. Findings from [16] indicate that the best performance can be achieved by including all information present in the test model. To access the distance between two objects in a multidimensional feature space, set-based similarity scoring mechanism can be used. In our case, a TC is not only a group of objects (e.g., states, transitions, input, output), it is also a sequence which could have a different order of these objects compared to other TCs. Therefore, also sequence-based distance algorithm can improve the test case selection. The result of the similarity computation is a similarity matrix which contains the pairwise similarity between all TCs. In the last step, the target number of TCs is selected based on the similarity matrix. It is the goal to find the group of TCs which have the lowest average similarity between the TCs of the selected group. Note, that this is a NP-hard problem [2] and the optimum can therefore not be found in polynomial time. In our example, discussed in Section IV-A, with a total number of n = 132 TCs the maximum number of combination possibilities is reached with a group size of k = 66 ($\binom{n}{k} \approx 3,8 \cdot 10^{38}$ with n=132 and k=66).

The presented work evaluates the performance of one set-based and a sequence-based similarity scoring computation together with a baseline random-search and Group Hill Climbing. The algorithms are outlined within this section, including pseudo code to ensure that the realisation approach can be validated and compared to other approaches.

A. Similarity Score Computation

As a baseline, a Hamming-inspired algorithm is realised according to the work discussed in [16]. The Hamming Distance is an edit-based distance algorithm, which is widely used in the literature. It defines the minimum required operations to transform one string to another with editing operations (delete, insert, substitute) for strings with the same length [17].

```
Input: allowedSymbols
foreach testCase do
    foreach allowedSymbol do
        if testCase contains allowedSymbol then
            occurrenceBit = 1
        else
            occurrenceBit = 0
        end
        Add occurrenceBit to bitOcurrenceStream
    end
end
foreach bitOcurrenceSteam do
    simStream = pairwise XOR of bitOcurrenceStream
    similarity = simStream / length of bitOccurenceStream
end
```

Figure 1. Hamming-based Similarity Scoring Algorithm.

The pseudo code shown in Figure 1, shows the basic steps for this approach. In all discussed similarity scoring mechanism, the algorithm starts with the identification of the allowed symbols. Dependent on the encoding, this could include input, output symbols, states, guards etc. For the sake of simplicity, all experiments are conducted with allowed symbols for input, output and states. Guards have not been taken into account (cf. Section IV-A which describes the example service). Although, if the algorithm extension would use also guards, it would not change the algorithm but would result in a larger symbol space, which consequences in lower similarities between all TCs. For each TC, the occurrence of the individual allowed symbols is identified. Each occurrence is documented with a *occurenceBit* = 1 and 0 if it is not present. Afterwards this *occurenceBit* is added to a *bitOccurenceStream*. For each TC the *bitOccurenceStream* is pair- and bitwise XOR compared to each *bitOccurenceStream* from all other TCs. The resulting XOR distance is then used to count the bits that are 1 and divide them by the length of the *bitOccurenceStream*. The result is stored as the pairwise similarity score. The classical Hamming Distance algorithm is limited to strings with the same length. Since TCs does not necessarily have the same

length, the classical algorithm has been altered as proposed by [11]. As a consequence the implemented Hamming inspired algorithm can handle strings with different length but is only set-based and does not take the occurrence order of the symbols into account.

```
Input: SymbolsOfEachCase
foreach pair of testCases do
    foreach symbolNum of FirstTestCase do
        m[symbolNum, 0] = symbolNum
    end
    foreach symbolNum of SecondTestCase do
        m[0, symbolNum] = symbolNum
    end
    foreach symbol of FirstTestCase do
        foreach symbol of SecondTestCase do
            if symbolFirst == symbolSecond then
                m[symbolFirst, symbolSec] = m[symbolFirst -1 , symbol-
                Sec -1]
            else
                m[symbolFirst, symbolSec] = Min (
                m[symbolFirst -1 , symbolSec] + 1,
                m[symbolFirst, symbolSec -1] + 1,
                m[symbolFirst -1 , symbolSec -1] + 1 )
            end
        end
    end
    similarity = 1 - m[numOfSymbolsFirst, numOfSymbolsSec] /maxNum-
    Symbols;
end
```

Figure 2. Levensthein-based Similarity Scoring based on Wagner-Fischer Algorithm [18].

The realisation of the Levenshtein similarity computation is shown in Figure 2. Levenshtein is also an edit-based distance algorithm and is not limited to strings with the same length [17]. Each edit operation (delete, etc.) results in an increasing distance between the compared strings. The Levenshtein algorithm is initiated by the identification of the symbols, which are part of the current pair of TCs. A matrix is created where the first row and column contain an increasing sequence from one to the quantity of symbols with an increment of one. Based on the basic operations of *add, del, modify* the sequence of symbols of the *FirstTestCase* is compared to the sequence of symbols of the *SecondTestCase*. It is identified how many operation steps are required to alter the *FirstTestCase* and reach the second one. The algorithm is based on the Wagner-Fischer algorithm [18] and it always prefers matches over insertions or deletions even if they provide a better score. The last computed matrix element contain the distance between the two TCs. For comparison reason this distance is then converted to a similarity score and normalised to the *maxNumSymbols*. Different to the realisation of Hematie et. al where only matches have been counted, this algorithm is able to identify the distance between two TCs as intended by the Levenshtein algorithm.

B. TC Selection

As the baseline, the random selection of a group of TCs has been implemented. As outlined in Figure 3 it starts with the input of the *Similarity Scores* matrix, all *TCs*, the target number of TCs and the number of trials (*numTrials*). The output of the algorithm is the list of selected TCs for the execution. For each trial, the target number of TCs is selected out of all TCs. Afterwards, the summary similarity between all selected TCs is computed based on the pairwise similarity scores. For each iteration, the computed summary similarity is compared to the lowest previous summary similarity. After the defined number of trials the group of TCs with the lowest summary similarity is selected for the test execution. While this approach finds the best group of TCs if the number of trials is infinity it is the question if other search algorithms exist, which can outperform this simple random methodology by identifying a better group of TCs within the same amount of resources (e.g., computation time).

```
Input: Similarity Scores
Input: Test Cases (TCs)
Input: Trial Numbers (numTrials)
Input: n /* Target number Test Cases                    */
Output: selTCs /* List of Selected TCs                  */
for trialNum = 0 to numTrials do
    tmpSelTCs /* list of selected TCs                   */
    tmpSelTCs = randomly select n test cases out of all
    foreach Selected Test Cases do
        rowSim /* Current Row Similarity                */
        rowSim = sum similarity of tmpSelTCs
        sumSim = sumSim + rowSim
    end
    lastSumSimilarity = 0
    if sumSim >= lastSumSimilarity then
        selTCs /* list of selected TCs                  */
        selTCs = tmpSelTCs
        lastSumSimilarity = sumSim
    end
end
```

Figure 3. Random Search Test Case Group Selection.

One promising candidate is Hill Climbing, which is based on the identification of local optimums. Instead of searching for the best possible solution out of all groups of TCs, it can help to find good but not necessarily the best group of TCs (local optimum). Hill Climbing searches for local optimums by looking at neighbour elements. A neighbour is an element which is structural close. From a start point, the algorithm tries to find a better solution by exchanging one element with a neighbour. The algorithm stops if no better neighbour is found (local optimum reached). Hill Climbing approaches differs how the starting point is chosen, how neighbours are defined and how many elements can be reached with one move operation (numbers of direct neighbours) [16].

The following application of a Group Hill Climbing is shown in Figure 4. The first step is equal to the Random Search algorithm with a number of trials of one. Then, for each TC within this group it is investigated if a neighbour TC exists, which lowers the summary similarity between all TCs of the selected group (*sumSim* $<$ *lastSumSimilarity*). The order of the created TCs thereby defines which TCs are neighbours. Therefore, the methodology how the TCs are created is expected to have influences on the performance of the identification of neighbours. The experiments are only conducted with the W-method and further investigations could quantify the influences of different test case creation methodologies. To reduce the computation effort only the variable part of the summary similarity is computed. The variable part of this summary is the pairwise similarity score between the selected group of TCs (without the current TC) to the neighbours of the current TCs (*rowSim*). The neighbour search is stopped as soon as there is a neighbour with worse characteristics than the last one. Therefore, the computation effort is not deterministic and

depends on the initial set of TCs and the characteristics of the neighbours (e.g., if the next optimum is nearby).

```
Input: Similarity Scores
Input: Test Cases (TCs)
Input: n /* Target number Test Cases                    */
selTCs /* list of selected test cases                   */
selTCs = randomly select n test cases out of all
foreach Selected TC do
    rowSim /* Current Row Similarity                    */
    rowSim = sum similarity of selTCs
    tmpSelTCs /* temp. list of sel. TCs                 */
    tmpSelTCs = selTCs - currentTestCase
    rTC /* Right TC of Current TC                       */
    rTC = currentTestCaseNumber + 1
    lTC /* Left TC of Current TC                        */
    lTC = currentTestCaseNumber - 1
    rRowSim /* Row Similarity with rTC                  */
    rRowSim = sum Similarity of rTC to tmpSelTCs
    lRowSim /* Row Similarity with lTC                  */
    lRowSim = sum similarity of lTC to tmpSelTCs
    if rowSim > rRowSim < leftRowSimiarity then
        laRowSim /* Last Row Similarity                */
        laRowSim = rRowSim
        while rowSim > laRowSim do
            laRowSim = rowSim
            rTC = rTC + 1
            rowSim = sum similarity of rTC to selTCs
        end
        tmpSelTCs = selTCs + (rTC - 1)
    else if rowSim > lRowSim < rRowSim then
        search left in the same way as befor for the right sight
    else
        tmpSelTCs = selTCs + currentTestCase
end
```

Figure 4. Hill Climbing Test Case Group Selection.

IV. Evaluation Results

The different methodologies to compute the similarity between the TCs as well as the minimisation algorithm results in several experimentation setups where the different aspects are evaluated individually. The next Subsection will briefly explain the overall test approach and introduces an example finite state machine, which is conducted for the experiment. Afterwards, the different experiments are explained and the results are discussed.

A. Setup

The TC selection concept of the present work is included in a test automation framework for IoT-based services. The framework is capable to create and execute IoT-domain specific TCs. Nevertheless, the presented test derivation concept is not limited to that and will be discussed here on a generalised level. The test creation process can be described as follows: the framework explained in detail in [19] derives a test model out of semantic descriptions of the SUT. Subsequently, the test model is represented by an extended finite state machine. This test model is utilised to derive TCs based on the W-method. Afterwards, the TCs are translated with a template engine into the TTCN-3 language which afterwards can be compiled and executed with the TTworkbench [20].

An example of a state machine with 5 states, 10 transitions, 2 input messages and 2 output messages is implemented and as a result the W-method creates 132 TCs with full state and transition coverage. The example state machine is depicted in Figure 5. It shows a behaviour of a reactive

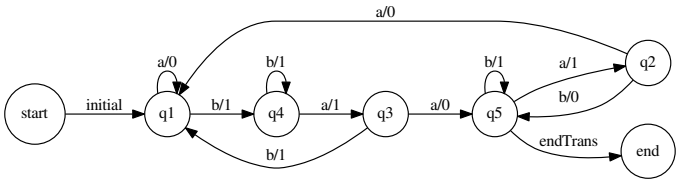

Figure 5. Example Finite State Machine.

system, which will be utilised to evaluate the performance of the similarity scoring and minimisation methodologies within this paper. As an example the input symbols are [*a,b*] and output symbols are [*0,1*]. In order to be executable in our test framework, also transitions for the initialisation (e.g., starting of the SUT) and an end transition exists but does not affect the experiments. The outlined experiments, shown during the next subsection, start after the derivation of the TCs. For each setup, the experiment has been repeated 10,000 times to ensure convincing results. For each combination, of similarity score computation and minimisation algorithm, the experiment is repeated with different target number of TCs from 130 to 5 TCs. Due to computation time the experiments are conducted with a step range of 10 between 130 and 10 target TCs. Although, the results follow a continuous curve and it is not expected to have a divergent behaviour between these values. Each experiment is shown with boxplots to visualise the distribution of the 10,000 repetitions of the experiment. In addition, the median average similarity score is shown in comparison to the other combinations of the algorithms. The time measurements indicate the implemented computation effort, although it does not replace theoretical analysis of the algorithm complexity. The boxplot whiskers show the lowest datum still within 1.5 Inter-Quartile Range (IQR) of the lower quartile and the highest datum still within 1.5 IQR of the upper quartile. Outliers are indicated with a circle.

B. Random Search and Group Hill Climbing with Hamming Similarity Scoring

Figure 6 shows the boxplots of the average similarity between the selected TCs as a function of the number of selected TCs (target TC number). The experiment is conducted with *N=1*, where *N* is the number of trials and with the Hamming Distance inspired scoring methodology. As a general characteristic, the median average similarity decreases while the number of selected TCs also decreases. The diversity of the results increases with the decreasing number of selected TCs due to larger influences of individual TCs, which can result in either small or large average similarity scores. The influence of the number of trials is shown in Figure 7, where the number of trials is N=100. Compared to *N=1* the diversity is much lower and reduces the median average similarity up to 0.067 with the number of selected TCs equals five.

Figure 8 shows the measurements conducted with the Group Hill Climbing algorithm. While the Group Hill Climbing can outperform random search with N=100 during 120 and 30 TCs, it shows limitations for numbers of selected TCs lower than 20. One reason is that the number of possible groups decreases for groups smaller than 66 (for this example) and thus helps the random search approach. As shown in Figure 9 Group Hill Climbing reduces the median average

 ISBN: 978-1-61208-370-4

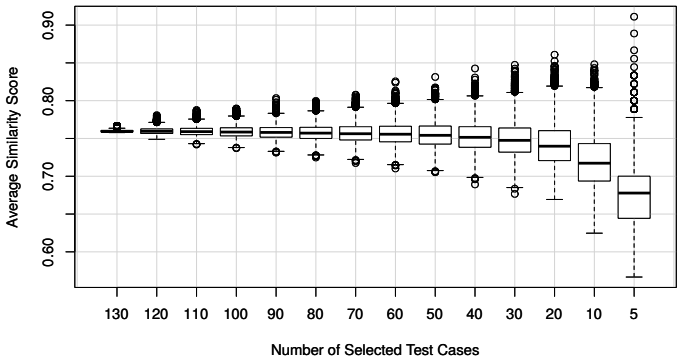

Figure 6. Boxplot of the Average Similarity Between the Selected TCs with Hamming Distance Inspired Scoring and Random Search with N=1.

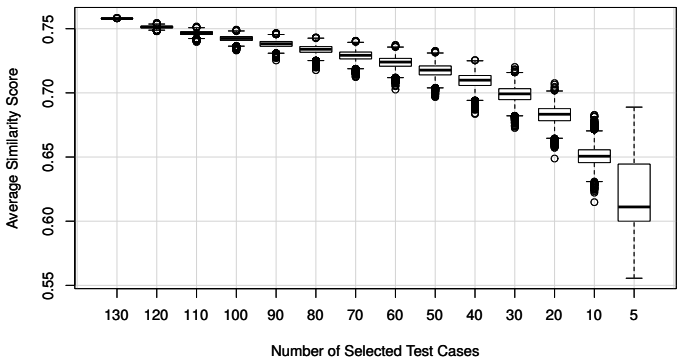

Figure 7. Boxplot of the Average Similarity Between the Selected TCs with Hamming Distance Inspired Scoring and Random Search with N=100.

similarity compared to Random Search with N=100 up to 0.02 (3,4%) with 70 selected TCs and up to 0,056 (6,6%) compared with Random Search with N=1. One explanation why the outperform is maximised with 60 and 70 TCs selected, is that the number of possible groups is maximised with a group size of 66 ($\binom{n}{k} \approx 3,8 \cdot 10^{38}$ with n=132 and k=66) and therefore random search is very limited. At the same time there are direct neighbours that have not been selected in the initial group and therefore group hill climbing can benefit from the correlation between closely TC neighbours. The computation effort of the implementation is shown in Figure 10. It indicates that Random Search with N=100 requires more computation time than Group Hill Climbing. The Group Hill Climbing computation time decreases with the number of selected TCs since only neighbours for each selected TC are identified. The Group Hill Climbing could be improved further (in terms of computation time), if the Group Hill Climbing tries to find either the group of TC which are not selected (by finding the maximised average similarity) or discover the group of TC which are selected (discover the minimum average similarity). The algorithm can always be applied to the smaller of these two groups and this would optimise the execution time. With this approach it can be assumed that the curve is mirror-symmetric to the right side of 66 selected TCs. Note, that the shown performance is dependent on the derived TCs and further investigations are required to verify this behaviour on different SUTs and different numbers of initial TCs.

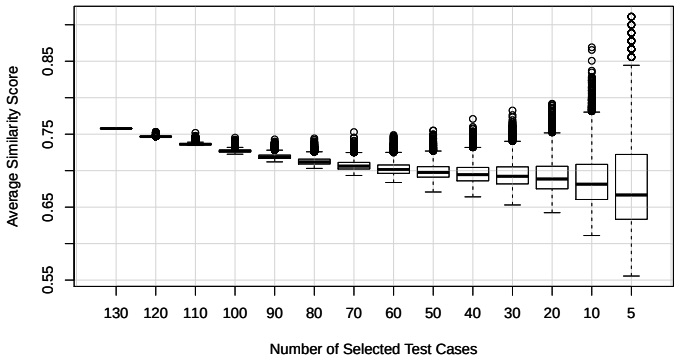

Figure 8. Boxplot of the Average Similarity Between the Selected TCs with Hamming Distance Inspired Scoring and Group Hill Climbing with N=100.

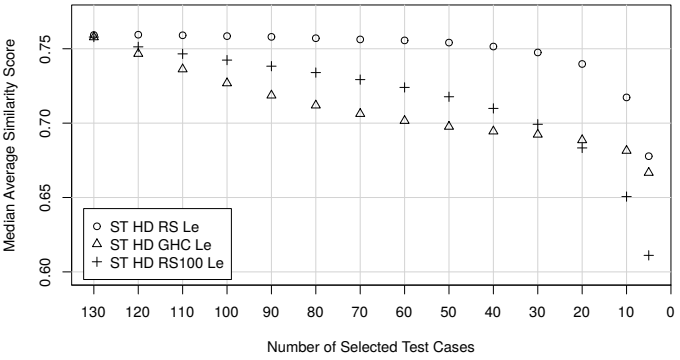

Figure 9. Average Median Similarity Between the Selected TCs with Hamming Distance Inspired Scoring and Group Hill Climbing Compared with Random Search.

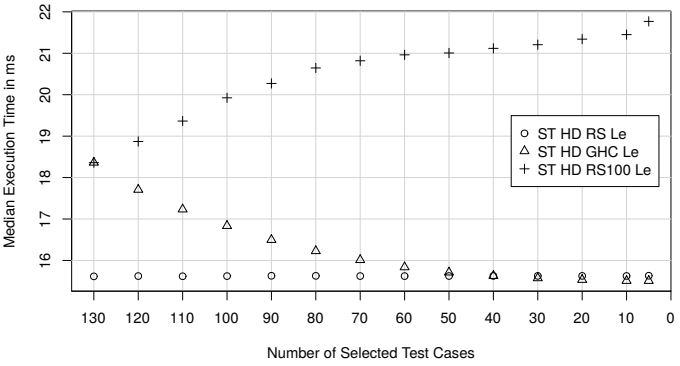

Figure 10. Median Execution Time as a Function of the Number of Selected TCs with Hamming Distance Inspired Scoring and Group Hill Climbing and Random Search.

C. Random Search and Group Hill Climbing with Levenshtein Scoring

As presented before the experiments have been repeated with the Levenshtein similarity scoring discussed in Section III-A. Figure 11 shows the random search with N=1 where N is the number of trials. The results can be compared to the results with Hamming-based Similarity Scoring. While the average similarity with 130 TCs starts with 0,55 instead of 0,76 (due to different normalisations), the trend with a decreasing number of TCs remains the same. The median average similarity decreases, while the diversity increases.

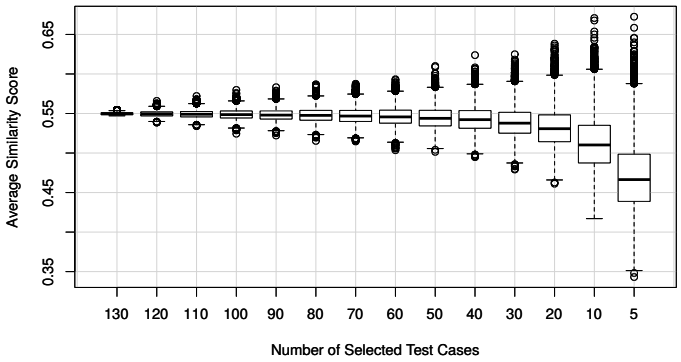

Figure 11. Boxplot of the Average Similarity Between the Selected TCs with Levenshtein Scoring and Random Search with N=1.

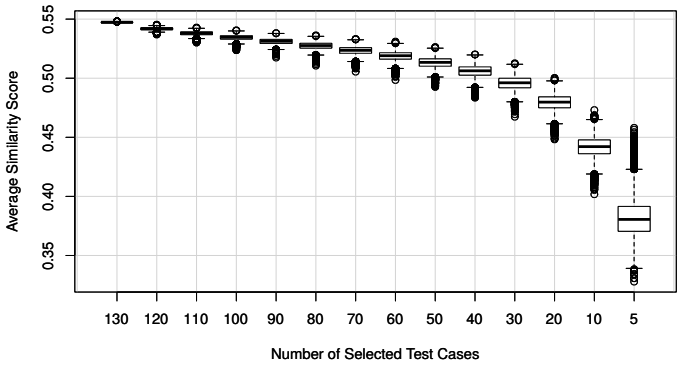

Figure 12. Boxplot of the Average Similarity Between the Selected TCs with Levenshtein Scoring and Random Search with N=100.

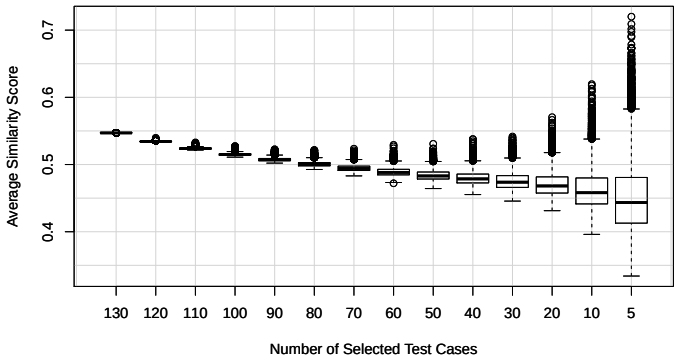

Figure 13. Boxplot of the Average Similarity Between the Selected TCs with Levenshtein Scoring and Group Hill Climbing.

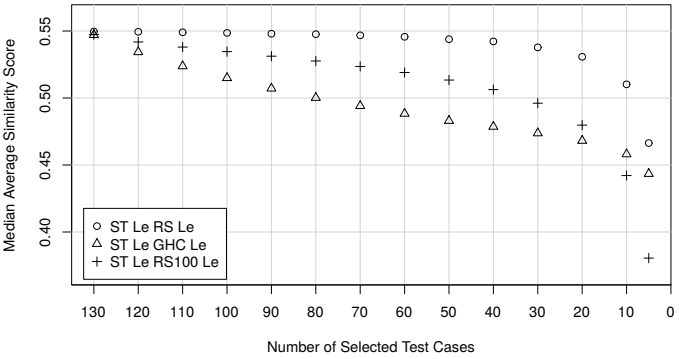

Figure 14. Average Median Similarity Between the Selected TCs with Levenshtein Inspired Scoring and Group Hill Climbing Compared with Random Search.

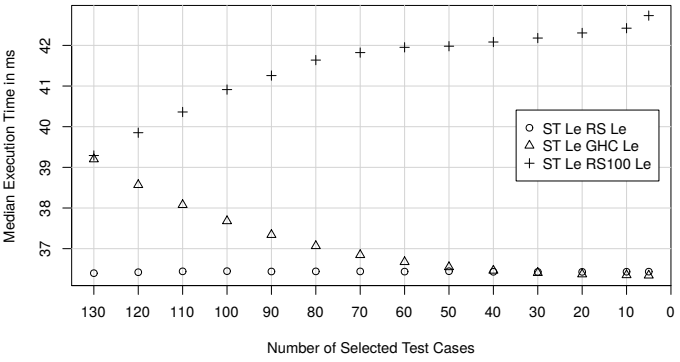

Figure 15. Median Execution Time as a Function of the Number of Selected TCs with Levenshtein and Group Hill Climbing and Random Search.

Also the enhancement of number of trials, shown in Figure 12 is comparable to the results with Hamming-based scoring. Figure 13 shows the average similarity score as the function of the number of TCs with Levenshtein Similarity Scoring. While the median average similarity decreases with a decreasing number of selected TCs, it exposes also the drawback of Group Hill Climbing, since the diversity of the average similarity score increases. It is much more likely to compute values that are worse than randomly generated with numbers of selected TCs from 20 to 5. As shown in Figure 14 Group Hill Climbing outperforms Random Search (max 0,03 (6%) for N=100 and 0,06 (10,9%) for N=1 for 70 TCs) between 30 and 120 TCs, while using less computation time (Figure 15). Although, the performance gain of Group Hill Climbing is higher with Levenshtein Similarity Scoring, the median execution time is higher compared with Hamming-based Similarity Scoring since the scoring mechanism requires more computation time. Therefore, future work will include the evaluation if Random-search with Hamming-based Scoring, with comparable computation time (more than 100 trials), can outperform the Group Hill Climbing performance with Levenshtein Similarity Scoring.

V. CONCLUSION

TC reduction based on diversity investigation is a promising approach to enable scalable model-based test automation. To enhance the understanding how distance and search-based minimisation algorithm can be correctly applied to select a group of TCs, Hamming-based Similarity Scoring and Levenshtein Similarity Scoring are evaluated together with random search and a proposed Group Hill Climbing algorithm. Based on an example finite state machine, experiments are conducted. The results indicate, that the best performance can be achieved with Group Hill Climbing and Levenshtein Similarity Scoring and on the same time it consumes less computation time compared to Random Search with 100 trials. Future work will include additional search- and similarity computation algorithms, which will be evaluated with different SUTs in order to generalise the results.

 ISBN: 978-1-61208-370-4

VI. ACKNOWLEDGEMENT

The research leading to these results has received funding from the European Union Seventh Framework Programme under grant agreement n° 257521.

REFERENCES

[1] E. G. Cartaxo, P. D. Machado, and F. G. O. Neto, "On the use of a similarity function for test case selection in the context of model-based testing," Software Testing, Verification and Reliability, vol. 21, no. 2, Jun. 2011, pp. 75–100.

[2] A. P. Mathur, Foundations of Software Testing. Addison-Wesley Professional, 2008.

[3] A. Gargantini, "4 conformance testing," in Model-Based Testing of Reactive Systems (Lecture Notes in Computer Science, vol. 3472), Broy M, Jonsson B, Katoen J-P, Leucker M, Pretschner A (eds). Springer: Berlin, 2005, pp. 87–111.

[4] W. Eric Wong, V. Debroy, and B. Choi, "A family of code coverage-based heuristics for effective fault localization," Journal of Systems and Software, vol. 83, no. 2, Feb. 2010, pp. 188–208.

[5] E. Engstrm, P. Runeson, and M. Skoglund, "A systematic review on regression test selection techniques," Information and Software Technology, vol. 52, no. 1, Jan. 2010, pp. 14–30.

[6] W.-T. Tsai, X. Zhou, Y. Chen, and X. Bai, "On testing and evaluating service-oriented software," Computer, vol. 41, no. 8, Aug. 2008, pp. 40–46.

[7] L. Zhang, D. Marinov, and S. Khurshid, "Faster mutation testing inspired by test prioritization and reduction," in Proceedings of the 2013 International Symposium on Software Testing and Analysis. ACM, Jul. 2013, pp. 235–245.

[8] C. Jard and T. Jéron, "Tgv: theory, principles and algorithms," International Journal on Software Tools for Technology Transfer, vol. 7, no. 4, Aug. 2005, pp. 297–315.

[9] H.-Y. Hsu and A. Orso, "Mints: A general framework and tool for supporting test-suite minimization," in Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on. IEEE, May 2009, pp. 419–429.

[10] F. Basanieri, A. Bertolino, and E. Marchetti, "The cow_suite approach to planning and deriving test suites in uml projects," in UML 2002The Unified Modeling Language. Springer, Sep. 2002, pp. 383–397.

[11] M. Harman, S. A. Mansouri, and Y. Zhang, "Search based software engineering: A comprehensive analysis and review of trends techniques and applications," Department of Computer Science, Kings College London, Tech. Rep. TR-09-03, Apr. 2009.

[12] S. Elbaum, A. Malishevsky, and G. Rothermel, "Incorporating varying test costs and fault severities into test case prioritization," in Proceedings of the 23rd International Conference on Software Engineering. IEEE Computer Society, May 2001, pp. 329–338.

[13] S. Yoo, M. Harman, P. Tonella, and A. Susi, "Clustering test cases to achieve effective and scalable prioritisation incorporating expert knowledge," in Proceedings of the eighteenth international symposium on Software testing and analysis. ACM, Jul. 2009, pp. 201–212.

[14] D. Leon and A. Podgurski, "A comparison of coverage-based and distribution-based techniques for filtering and prioritizing test cases," in Software Reliability Engineering, 2003. ISSRE 2003. 14th International Symposium on. IEEE, Nov. 2003, pp. 442–453.

[15] E. G. Cartaxo, F. G. O. Neto, and P. D. Machado, "Automated test case selection based on a similarity function." GI Jahrestagung (2), vol. 7, Sep. 2007, pp. 399–404.

[16] H. Hemmati, A. Arcuri, and L. Briand, "Achieving scalable model-based testing through test case diversity," ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 22, no. 1, Feb. 2013, p. 6.

[17] D. Gusfield, Algorithms on strings, trees and sequences: computer science and computational biology. Cambridge University Press, 1997.

[18] R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," Journal of the ACM (JACM), vol. 21, no. 1, Jan. 1974, pp. 168–173.

[19] D. Kuemper, E. Reetz, and R. Tonjes, "Test derivation for semantically described iot services," in Future Network and Mobile Summit (FutureNetworkSummit), 2013. IEEE, Jul. 2013, pp. 1–10.

[20] Testing Technologies, "TTworkbench," Website, Aug 2014, available online at http://www.testingtech.com/products/ttworkbench.php retrieved: Aug. 2014.

 ISBN: 978-1-61208-370-4