Pareto efficient multi-objective black-box test case selection for simulation-based testing

Aitor Arrieta[a], Shuai Wang[b], Urtzi Markiegi[a], Ainhoa Arruabarrena[a], Leire Etxeberria[a], Goiuria Sagardui[a]

[a] *Mondragon Unibertsitatea, Mondragon, Spain*
[b] *Testify AS, Oslo, Norway*

Abstract

Context: In many domains, engineers build simulation models (e.g., Simulink) before developing code to simulate the behavior of complex systems (e.g., Cyber-Physical Systems). Those models are commonly heavy to simulate which makes it difficult to execute the entire test suite. Furthermore, it is often difficult to measure white-box coverage of test cases when employing such models. In addition, the historical data related to failures might not be available.

Objective: The objective of the approach presented in this paper is to cost-effectively select test cases without making use of white-box coverage information or historical data related to fault detection.

Method: We propose a cost-effective approach for test case selection that relies on black-box data related to inputs and outputs of the system. The approach defines in total six effectiveness measures and one cost measure followed by deriving in total 21 objective combinations and integrating them within Non-Dominated Sorting Genetic Algorithm-II (NSGA-II). The proposed six effectiveness metrics are specific to simulation models and are based on anti-patterns and similarity measures.

Results: We empirically evaluated our approach with these 21 combinations using six case studies by employing mutation testing to assess the fault revealing capability. We compared our approach with Random Search (RS), two many-objective algorithm, as well as three white-box metrics. The results demonstrated that our approach managed to improve Random Search by up to around 28% in terms of the Hypervolume quality indicator. Similarly, black-box metrics-based test case selection also significantly outperformed those of white-box metrics.

Conclusion: We demonstrate that test case selection is a non-trivial problem in the context of simulation models. We also show that the proposed effectiveness metrics performed significantly better than traditional white-box metrics. Thus, we show that black-box test selection approaches are appropriate to solve the test case selection problem within simulation models.

Keywords: Test case selection, Search-based software engineering, Simulation-based testing, Cyber-Physical Systems

1. Introduction

Simulation models (e.g., Simulink) are commonly employed to model and simulate complex systems, such as Cyber-Physical Systems (CPSs). In such systems, where the software closely interacts with physical processes, simulation models are employed in early stages before the system is implemented [1]. Simulation models permit raising the level of abstraction of complex systems at which testing is performed by representing relevant aspects of system behavior, structure, properties or the environment [2]. These models are typically used in several domains, including the automotive [1, 3, 4] and the vertical transport domain [5].

Testing these models can be time-consuming due to several reasons. First, the systems being tested using simulation models are usually tested at system level, which makes the test execution time of a test case longer compared to unit testing [6]. Secondly, in the case of CPSs, the physical layer is often modeled with complex mathematical models, which makes the simulation time lengthy [7]. Third, simulation models might be tested at different test levels, such as Model-, Software- and Hardware-in-the-Loop (MiL, SiL and HiL) [7]. Specifically, the HiL test level is a real-time simulation and it is practically infeasible to execute a large amount of test cases [7]. Fourth, simulation of complex systems might be driven by co-simulation. This paradigm aims at connecting different simulation tools enabling engineers a higher flexibility. However, exchanging data among the different simulation tools and synchronizing the clocks leads to a higher test execution time.

The simulation cost of a simulation model depends on several factors. First, the **fidelity of the model** considerably influences the test suite execution cost. Having a low fidelity model can be appropriate to test some functionalities in early verification stages. However, if the fidelity

Email addresses: aarrieta@mondragon.edu (Aitor Arrieta), shuai.wang@testify.no (Shuai Wang), umarkiegi@mondragon.edu (Urtzi Markiegi), eibek03@mondragon.edu (Ainhoa Arruabarrena), letxeberria@mondragon.edu (Leire Etxeberria), gsagardui@mondragon.edu (Goiuria Sagardui)

of the model is very low, the model will be far from reality and it will not let engineers test many properties. For instance, based on our experience of collaborating with electrical engine developers, with low fidelity models they cannot model the temperature of the engines with high precision. Thus, all functionalities related to this property cannot be appropriately tested in some applications where this is critical (e.g., applications of the railway domain). Conversely, having a high fidelity model can be beneficial as the simulation can be more realistic and thus, many properties can be tested. Nevertheless, having a high fidelity model can result in the drawback of requiring too complex mathematical models in order to model the system, and thus, dramatically increases the simulation time. Second, the **simulation time step** has an influence on the cost of the execution of test cases [8]. The simulation of a model is performed in a finite number of steps. A lower simulation time step means a more realistic simulation, but more steps will be required to be simulated, increasing the simulation time. Third, **the test level in which the simulations are performed** have a strong influence on the time [7, 9, 10]. When testing complex systems such as CPSs, several test levels are employed, named as Model-, Software-, Processor- and Hardware-in-the-loop (MiL, SiL, PiL and HiL). The tests are executed in each of the levels several times. Some test levels, e.g., the HiL test level is a real-time simulation (further details in Section 2.2). Executing many test cases at this level will be very time consuming. Typically, executing test cases at other levels, e.g., at MiL or SiL is faster, although in several cases, if a very high fidelity model is employed, the simulation time is lengthier than at the HiL test level. The last part is related to the **semantics of the simulation models**, which also have an impact on the simulation time (e.g., when Functional Mock-up Units (FMUs) are used, the simulation time is typically faster, but the generation of these FMU-s is not always available).

In practice, when real industrial systems are employed, the execution of test suites is lengthy. When considering a case study from our industrial partner, where a high fidelity co-simulation model of an elevator was used [11], executing a simulation of 20 seconds took us around 1 minute. Besides, when testing these systems much longer test cases are required (e.g., simulations of full office days (i.e., 10 to 12 hours in virtual time) or worst case scenarios of Up and Down-peak traffic in elevator systems that take around 2 hours (in virtual time)). In another industrial system from the satellite domain, executing 20 realistic test cases by employing co-simulation took 29,688 seconds (i.e., around 8.2 hours) [12].

As a result, selecting the appropriate subset of test cases to execute the test cases in a cost-effective manner is crucial. In the current state-of-the-art, several approaches have focused on selecting test cases with the objective of increasing white-box coverage [13, 14]. However, obtaining information related to white-box coverage in these cases is often a challenge. First, CPSs integrate digital cyber computations with physical processes. While obtaining information of white-box coverage related to the cyber layer might be trivial because the model is discrete, the coverage related to the physical layer might be infeasible to obtain as their models are continuous models. Secondly, current approaches for modeling and simulating CPSs are acquiring the Functional Mockup Interface (FMI) standard [15]. This standard aims at interchanging different models of different simulation tools and vendors by creating an FMU. The FMUs are executable black-box units that do not permit measuring internal coverage. Other approaches have taken advantage of using historical data related to failures to make the test selection more precise [13, 7, 16]. However, to obtain historical data related to failures, tests need to be executed several times.

To address the above-mentioned challenges, we propose a multi-objective test case selection approach that relies on black-box data. We define six effectiveness measures by identifying and analyzing four different anti-patterns based on output signals (instability, discontinuity, growth to infinity and output minimum and maximum difference) and two similarity metrics (input and output-based similarities). We integrated different objective combinations within the Non-Dominated Sorting Algorithm-II (NSGA-II), which has been successfully applied in software engineering [3, 4, 17], including other test case selection approaches [13, 18, 16]. We derived different fitness functions by using up to three objective functions in each. In all of them we considered the Test Execution Time (TET) as the cost measure. The maximum number of objective functions was 3 because, despite NSGA-II being successfully applied to solve software engineering problems, it suffers from scalability issues when solving optimization problems with more than three objectives [19]. Moreover, based on our experience of applying multi-objective search in practice, we observed that when the number of objectives are more than three, the practical cost (e.g., running time) can be increased exponentially. Thus, we derived in total 21 objective combinations (six of them considering the TET together with one of the six effectiveness measures and fifteen of them considering TET together with each pair of the six effectiveness measures).

Following this, we empirically evaluated all the objective combinations within NSGA-II and RS by considering six different case studies (one of which is an industrial case study) and mutation testing. Furthermore, we compared the proposed black-box metrics with three traditional white-box metrics (i.e., Decision Coverage (DC), Condition Coverage (CC) and Modified Condition/Decision coverage(MC/DC)) and NSGA-II. These coverage metrics have been widely applied for test selection purposes [20, 21, 22]. Results indicated that our approach was cost-effective when compared to RS in 5 out of 6 case studies, and managed to outperform RS by 26% on average using the widely applied Hypervolume (HV) quality indicator [17, 3, 4]. Furthermore, overall, the proposed black-box metrics significantly outperformed the selected white-box

metrics. As for the different objective combinations, the anti-patterns have performed significantly better than the defined similarity measures.

This paper is an extension of our previous GECCO 2018 paper [8]. We extended the paper from the following perspectives:

- We propose a new black-box objective function for test selection and include in total six more fitness combinations. This new objective function considers the difference between maximum and minimum values of each output signal of a model, and complements one of the anti-patterns proposed in the conference version paper [8]. Some of the derived six fitness combinations have shown good performance when compared to other combinations proposed in the original paper. In fact, for one of the evaluation metrics and one of the case studies, the proposed combination along with another anti-pattern (i.e., growth to infinity) performed best, having the best average values.

- We compare our approach with three traditional white-box objective functions (CC, DC and MC/DC Coverage). The empirical evaluation suggests that selecting test cases with black-box objective functions is better than employing white-box coverage metrics, especially when considering the number of detected faults.

- We include an additional evaluation metric to assess our approach in a more comprehensive way. The metric, named as Average Weighted Mutation Score and Normalized Test Execution Time (W_MS_TET) returns the average weighted mutation score and test execution time of each of the solutions in the Pareto-frontier, without making assumptions that effective decision-makers exist. Overall, our approach was better than RS and white-box metrics for this metric too.

- We include two new case studies, one of which is larger than the case studies used in the conference version paper.

- We include two many-objective algorithms in our evaluation (NSGA-III and MOEA/D) to solve the scalability problems given by NSGA-II when dealing with more than two objectives. The results showed that overall, MOEA/D performed better than NSGA-III and that these two algorithms performed better than when employing the NSGA-II and white-box metrics. However, selecting the appropriate effectiveness measure along with the selected cost measure and NSGA-II was better than employing a many-objective algorithm with many effectiveness measures.

- Lastly, we have significantly expanded the related work section and have included a background section.

The rest of the paper is structured as follows. Section 2 provides general theory related to simulation models, simulation-based testing and Pareto-based search algorithms. Section 3 explains our multi-objective approach for test case selection of simulation models. Section 4 presents the experiment design to evaluate our approach. Results for the performed experiments and their discussion are presented in Sections 5 and 6. Section 7 discusses the main threats of the approach and how we tried to mitigate them. Section 8 positions our approach with the literature. We conclude the paper and discuss future directions in Section 9.

2. Background

2.1. Simulation models and basic notation

Simulation is referred to the process of creating a digital model to predict its performance in the real world. Developers of CPSs heavily rely on simulation environments and Model-Based Design (MBD) work-flows, where graphical models of their systems are employed to do rapid prototyping [23]. CPS modeling and simulation tools (e.g., Simulink) are dataflow models, where each model contains a set of blocks [23]. Each block accepts data through its inputs and may pass output through its output ports after performing a set of operations (e.g., mathematical, logical, etc.). Consider the example of a simulation model depicted in Figure 1, which includes six inputs (enable, brake, set, speed, inc and dec) and two outputs (throt and target). These tools permit modeling the system at different hierarchical levels. For instance, in the example shown in Figure 1, the model has two hierarchical levels. This permits engineers to organize their model into different levels according to the complexity of the different subsystems.

Let $SM = (I, O)$ be a simulation model, where $I = \{i_1, i_2, ..., i_N\}$ is a subset of inputs and $O = \{o_1, o_2, ..., o_M\}$ is a subset of outputs [1]. Each input and output of the simulation model is a signal (i.e., a function of time), which is stored as a vector where elements are indexed by time [1]. The simulation time (T) is divided into a set of equal sample time steps (ΔT) [1]. A signal (sig) is a function in time of a set of k number of observed simulation steps (i.e., $sig : \{0, \Delta T, 2 \times \Delta T, ..., k \times \Delta T\}$). For instance, a simulation of 10 seconds (i.e., T=10), with a sample time of 0.05 seconds (i.e., $\Delta T = 0.05$) would have a total of 10/0.05+1 (i.e., k=201) simulation steps. The lower the sample time is, the higher the precision of the simulation. However, the time required to simulate the system will also be lengthier. In our study, for each model being tested, we consider a fixed simulation sample time (i.e., the same for all the test cases).

Let $TS = \{tc_1, tc_2, ...tc_{NTC}\}$ be a test suite of NTC number of test cases and each tc is a set of input signals ($ISig$) that stimulate the inputs i_1 to i_N of SM (i.e., $tc_j = \{Isig_{j_1}, Isig_{j_2}, ..., Isig_{j_N}\}$) [1]. For each test case, we can obtain the output signals ($OSig$) by

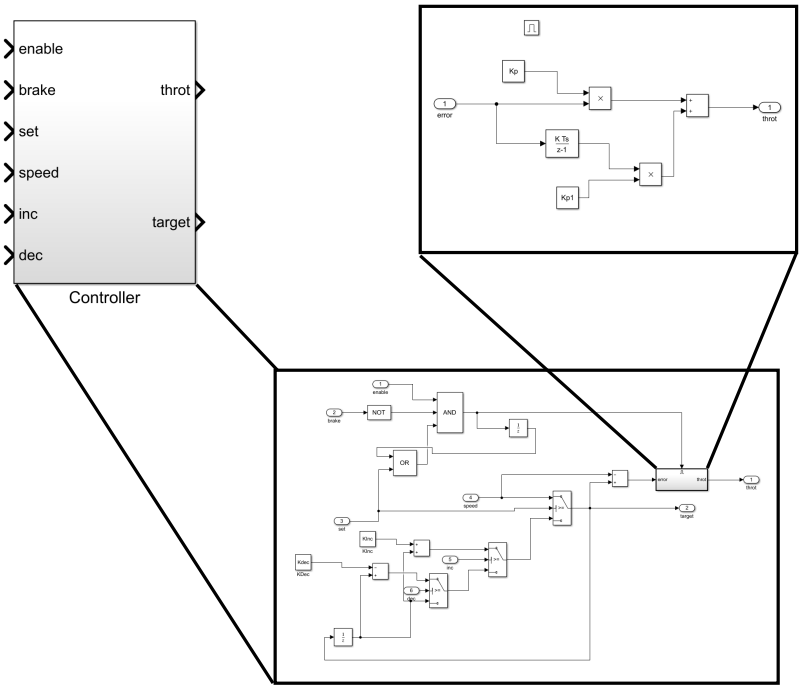

Figure 1: Example of a simulation model of a Cruise Controller of a car

simulating the simulation model SM, (i.e., $O\_tc_j = \{Osig_{j_1}, Osig_{j_2}, ..., Osig_{j_M}\}$). The goal of our approach is to consider the input and output signals of each test case to derive a new test suite (i.e., $TS_{sk} = \{tc'_1, tc'_2, ..., tc'_{NTC_{sk}}\}$) which cost-effectively tests SM such that $1 \leq NTC_{sk} \leq NTC$. However, the search space for test selection is huge. For example, suppose there are 100 test cases to test SM, then there will be $2^{100} - 1$ potential solutions for deriving the new test suite.

2.2. Simulation-based testing

The test execution of embedded systems and CPSs follow different steps and are performed at different test levels. These levels, coined as Model-in-the-Lopp (MiL), Software-in-the-Loop (SiL), Processor-in-the-Loop (PiL) and Hardware-in-the-Loop (HiL), are typically employed in the work-flow at the different verification and validation stages. At the first stage, when engineers are building the model of the CPS, the MiL test level is employed. MiL is the basic simulation that engineers employ to analyse the embedded software's model with the physical plant of the CPS [24]. At this level, the software is a model and the computations are performed using floating-point arithmetic to obtain the simulation results as reference values [24]. These reference values are later used and compared in the following simulations. When the test results are considered satisfactory at the MiL test level, the code of the controller is generated and the software's model is replaced with an executable code (e.g., a *.dll) [24]. At this test level, fixed-point computations are employed, as would be done in the real target processor. At the following test level (i.e., PiL), the code is cross-complied and executed on the real target processor, which communicates with the simulation tool that simulates the physical layer of the CPS in the computer [24]. The main objective of this test level is to detect potential faults introduced by the compiler [24]. In many occasions, the PiL test level is avoided with the HiL test level directly employed. In fact, from our interview with industrial partners, their aim is usually to start this phase as fast as possible. At the HiL test level, the embedded software is integrated with the Electronic Control Unit (ECU) and all the real-time infrastructure (including drivers, real-time operating systems, communications, etc.) [24]. The physical layer is encapsulated and emulated in an embedded device (e.g., FPGA) and the simulation is performed in real-time [24]. At this level, apart from functional requirements, non-functional requirements are also tested, as temporal constraints are critical when testing CPSs. There are several approaches for testing CPSs at different test levels, from MiL and SiL [25, 1, 26, 27, 28, 5, 11] to HiL [29, 10].

From our experience with industrial partners, it is noteworthy that test cases are not executed at each level only once, but several times. Furthermore, the software testing phases within these systems are carried out in a scaled way. The initial model is usually a low fidelity and simple model, but as functionalities start getting validated, simple subsystems of the CPS model start getting substituted by more complex ones. At the HiL test level this is similar. The ECU includes other functionalities and tasks

apart from the functionalities that are tested at the MiL and SiL test level. At this stage, tests are executed several times, as there are many changes required (e.g., priorities of tasks, functionalities not tested in previous stages are included, communication protocols, changes in the configuration of the operating system, etc.). Thus, at all test levels, regression test optimization approaches play an important role.

2.3. Pareto-based search algorithms

Many software engineering problems are multi-objective in nature, meaning that they contain multiple conflicting objectives to be optimized. For this reason, in the last few years, the use of Pareto-based multi and many-objective search algorithms has been investigated to solve Search-Based Software Engineering (SBSE) problems [30, 13, 31, 32, 3, 17, 33, 19, 34]. Unlike single-objective search algorithms, Pareto-based search algorithms produce a set of non-dominated solutions (also known as Pareto front), from which the user can select one or more solutions based on their specific needs [17].

The NSGA-II [35] is one of the most well-known multi-objective search algorithms. As common Genetic Algorithms (GAs), NSGA-II generates an initial set of random solutions. These solutions later evolve through a series of generations to find better solutions [19]. To this end, new solutions (i.e., offsprings) are created by using the crossover and mutation operators. To generate the offsprings, parents are selected with a selection operator, which uses Pareto optimality to give higher probability to select non-dominated solutions [19]. To preserve solutions forming the non-dominated solutions in the next generation, NSGA-II uses a fast non-dominated sort algorithm (i.e., elitism) [19]. NSGA-II has been applied in many SBSE problems (e.g., test case generation [3, 4, 36], test case selection [13, 16], test case prioritization [37], software effort estimation [33], etc.).

Apart from NSGA-II, other typical multi-objective algorithms include SPEA2 [38] and PESA-II [39]. However, these algorithms have shown scalability problems with more than three objectives [19]. To solve these scalability problems, in the last few years, many-objective algorithms have been proposed, which are designed for more than three objectives, including NSGA-III [40] and MOEA/D [41].

3. Multi-Objective Test Case Selection Approach

3.1. Cost Measure: Test Execution Time

In different studies, the number of test cases has been considered as the cost measure for minimizing the initial test suite (e.g., [42]). However, the time it takes a test case to be executed can differ and thus, we have considered the Test Execution Time (TET) of the entire test suite as the cost measure to guide the search. Thus, executing test cases with lower TET results in a lower cost. Given a solution sk with NTC_{sk} test cases, (i.e., $TS_{sk} = \{tc'_1, tc'_2, ..., tc'_{NTC_{sk}}\}$), the TET to execute the test suite is given in Equation 1, where ET_{tc_i} is the time it takes to execute the i-th test case in TS_{sk}. In this context, it is noteworthy that as the TET we consider the virtual time. This is because it is not trivial to obtain the real-time that a test case takes to execute, because it varies as the model evolves. Nevertheless, the difference on TET among test cases are proportional to the virtual time and thus, it is possible to consider the virtual time as the cost measure.

$$TET_{sk} = \sum_{i=1}^{NTC_{sk}} ET_{tc_i} \quad (1)$$

3.2. Effectiveness Measures

Six different effectiveness measures were defined, three of them (i.e., Instability, Discontinuity and Growth to Infinity (Figure 2)) were based on anti-patterns of simulation models that were identified in previous studies [43, 44]. One of the anti-patterns is proposed in this paper. The remaining two effectiveness measures are related to similarity measures. Notice that these metrics are not cumulative, unlike white-box metrics, where for instance, two test cases might cover 90% of coverage but are exercising the same paths and one test case covers only 10% but is exercising the paths that the previous test cases are not exercising. Thus, in terms of coverage, it would be better to select one of the test cases exercising 90% of the paths and the test case exercising the remaining 10% of the paths.

3.2.1. Instability

Instability refers to the anti-pattern where an output signal shows quick and frequent oscillations [44], as can be shown in Figure 2a. For instance, this anti-pattern can appear, when a state-chart controller quickly switches between states. The presence of this anti-pattern in simulation models can have an undesirable impact on physical processes [44]. The instability of a signal output sig is given in Equation 2, where k is the number of simulation steps and ΔT is the simulation time step. The higher the value of the function $instability(sig)$, the higher the instability degree of the signal.

$$instability(sig) = \sum_{i=1}^{k} |sig(i \cdot \Delta T) - sig((i-1) \cdot \Delta T)| \quad (2)$$

In our approach, to perform the test case selection by using this anti-pattern, we obtain an instability degree for each of the test cases in the initial test suite (i.e., TS). This is done by observing each output of the simulation model for each test case. However, since each output in SM can represent a particular variable in different units (e.g., one output can represent vehicle speed

in km/h while another output can represent the acceleration in m/s^2), it is mandatory to normalize the instability of each test case. Given an $SM = \{I, O\}$ with M outputs (i.e., $O = \{o_1, o_2, ..., o_M\}$), the instability of a test case j in TS is obtained with Equation 3, where $instability(Osig_{j_i})$ is the instability of the i-th signal for tc_j and $\max(instability(Osig_i))$ is the maximum instability degree obtained in the i-th signal when considering all test cases in TS.

$$TCInstability(tc_j) = \frac{\sum_{i=1}^{M} instability(Osig_{j_i})}{\max(instability(Osig_i))) \cdot M} \quad (3)$$

Similarly, given a solution sk with NTC_{sk} test cases (i.e., $TS_{sk} = \{tc'_1, tc'_2, ..., tc'_{NTC_{sk}}\}$), we define the overall instability of the test suite as the sum of the normalized instability of each selected test case, as given in Equation 4. A higher degree of instability in the test suite means a higher probability of finding a fault due to this anti-pattern.

$$TSInstability(TS_{sk}) = \sum_{i=1}^{NTC_{sk}} TCinstability(tc_i) \quad (4)$$

3.2.2. Discontinuity

Discontinuity is an anti-pattern in which an output signal shows a short duration pulse [44], as depicted in Figure 2b. The discontinuity of an output signal can be measured by computing its derivative function. However, since simulation time steps are not infinitesimal, derivatives cannot be computed [44]. Matinnejad et al. proposed a function that relies on discrete change rates [43]. Given a signal sig, $lc_i = |sig(i \cdot \Delta t) - sig((i - dt) \cdot \Delta t|/\Delta t$ is the left change rate at step i, and $lc_i = |sig((i+dt) \cdot \Delta t - sig(i \cdot \Delta t)|/\Delta t$ is the right change rate at step i [43, 44]. The continuity of a signal sig can be measured following Equation 5, which is defined as the maximum of the minimum of the left and right change rates at each simulation step for a given set of simulation steps [43, 44].

$$discontinuity(sig) = \max_{dt=1}^{3}(\max_{i=dt}^{k-dt}(\min(lc_i, rc_i))) \quad (5)$$

As in the case of the instability anti-pattern, when measuring the discontinuity degree of each test case, a normalization process is required when more than one output exists. Given an $SM = \{I, O\}$ with M outputs (i.e., $O = \{o_1, o_2, ..., o_M\}$), the discontinuity of a test case j in TS is obtained with Equation 6, where $discontinuity(Osig_{j_i})$ is the growth to infinity of the i-th signal for tc_j and $max(discontinuity(Osig_i))$ is the maximum derivative value in the i-th signal when considering all test cases in TS.

$$TCDiscontinuity(tc_j) = \frac{\sum_{i=1}^{M} discontinuity(Osig_{j_i})}{max(discontinuity(Osig_i))) \cdot M} \quad (6)$$

To measure the discontinuity degree of a solution sk returned by the search algorithm, where NTC_{sk} is the number of test cases (i.e., $TS_{sk} = \{tc'_1, tc'_2, ..., tc'_{NTC_{sk}}\}$), we define the overall discontinuity of the test suite as the sum of the normalized discontinuity of each selected test case (Equation 7). A higher degree of discontinuity in the test suite means a higher probability of finding a fault due to this anti-pattern.

$$TSDiscontinuity(TS_{sk}) = \sum_{i=1}^{NTC_{sk}} TCDiscontinuity(tc_i) \quad (7)$$

3.2.3. Growth to infinity

Another anti-pattern that can provide an effectiveness measure when selecting test cases can be the growth to infinity pattern. In this case, an output signal of the simulation model grows to an infinite value [43], as shown in Figure 2c. Growth to infinity of a signal sig is measured by obtaining its maximum absolute value in all sample steps (Equation 8).

$$inf(sig) = max|sig(i \cdot \Delta T)| \quad (8)$$

Similar to the remaining anti-patterns, to obtain a global growth to infinity degree of a test case, a normalization process is required. Given an $SM = \{I, O\}$ with M outputs (i.e., $O = \{o_1, o_2, ..., o_M\}$), the growth to infinity of a test case j in TS is obtained with Equation 9, where $inf(Osig_{j_i})$ is the growth to infinity of the i-th signal for tc_j and $max(inf(Osig_i))$ is the maximum infinity value in the i-th signal when considering all test cases in TS.

$$TCInfinity(tc_j) = \frac{\sum_{i=1}^{M} inf(Osig_{j_i})}{max(inf(Osig_i))) \cdot M} \quad (9)$$

To measure the growth to infinity degree of a solution sk returned by the search algorithm, where NTC_{sk} is the number of test cases (i.e., $TS_{sk} = \{tc'_1, tc'_2, ..., tc'_{NTC_{sk}}\}$), we define the overall infinity of the test suite as the sum of the normalized infinity of each selected test case (Equation 10). A higher degree of growth to infinity in the test suite means a higher probability of finding a fault due to this anti-pattern.

$$TSInfinity(TS_{sk}) = \sum_{i=1}^{NTC_{sk}} TCInfinity(tc_i) \quad (10)$$

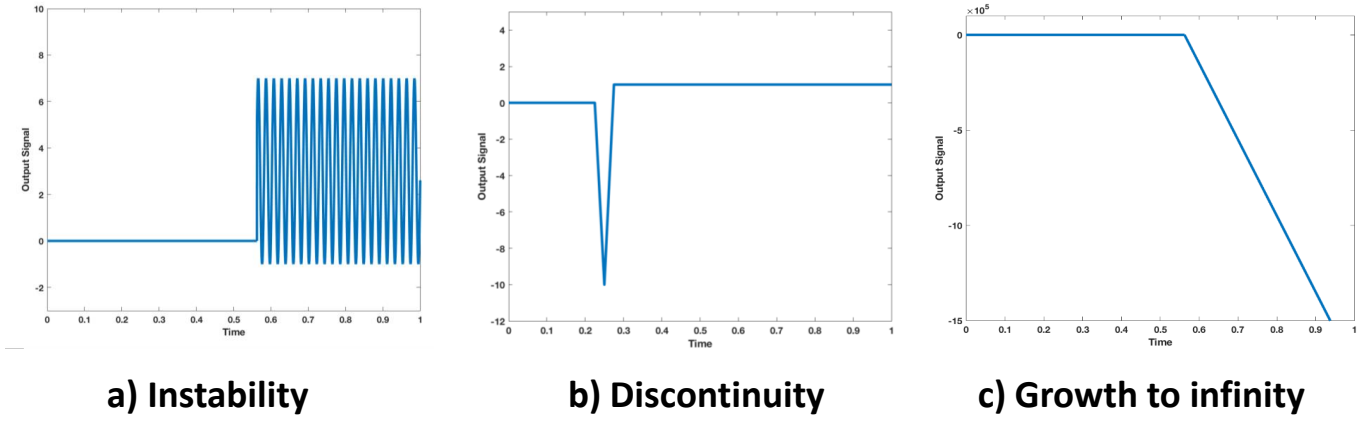

Figure 2: Anti-patterns for simulation models

3.2.4. Output minimum and maximum difference

In this paper we propose an additional effectiveness metric. In unit testing, another anti-pattern could occur if the output values exceed certain bounds. In the case of simulation-based testing, the output typically do not have these bounds (e.g., when simulating the position of a car or a drone). As an alternative, we propose effectiveness metric as the difference between maximum and minimum values that the outputs of the simulation model under test can have. This can also give a prediction of how throughly a model has been tested, because when small parts of a simulation model are tested, the outputs typically get stabilized on certain values. Furthermore, this new anti-pattern complements the growth to infinity anti-pattern presented in Section 3.2.3; while the growth to infinity anti-pattern considers only the the maximum absolute value of a specific signal, our metric considers the difference of these. A problem with the growth to infinity anti-pattern could be that in some simulations, the output values of a specific simulation model already begins from the maximum value. Thus, even if the signal does not have any variation, the anti-pattern will categorize it as an effective test case according to the growth to infinity anti-pattern. Conversely, in the case of this new metric, the variation is also considered. The output minimum and maximum difference of a signal *sig* is measured by obtaining its maximum value as well as its minimum value in all sample steps and subtracting these values (Equation 11).

$$difmaxmin(sig) = |max(sig(i \cdot \Delta T)) - min(sig(i \cdot \Delta T))| \tag{11}$$

To obtain a global output minimum and maximum difference of a test case, similarly to the previous anti-patterns, a normalization process is required. Given an $SM = \{I, O\}$ with M outputs (i.e., $O = \{o_1, o_2, ..., o_M\}$), the output minimum and maximum difference of a test case j in TS is obtained with Equation 12, where $difmaxmin(Osig_{j_i})$ is the growth to infinity of the i-th signal for tc_j and $difmaxmin(inf(Osig_i))$ is the maximum infinity value in the i-th signal when considering all test cases in TS.

$$TCdifmaxmin(tc_j) = \frac{\sum_{i=1}^{M} difmaxmin(Osig_{j_i})}{max(difmaxmin(Osig_i))) \cdot M} \tag{12}$$

To measure the output minimum and maximum difference degree of a solution sk returned by the search algorithm, where NTC_{sk} is the number of test cases (i.e., $TS_{sk} = \{tc'_1, tc'_2, ..., tc'_{NTC_{sk}}\}$), we define the overall output minimum and maximum difference of the test suite as the sum of the normalized output minimum and maximum difference of each selected test case (Equation 13). A higher degree of output minimum and maximum difference in the test suite means a higher probability of finding a fault due to this anti-pattern.

$$TSdifmaxmin(TS_{sk}) = \sum_{i=1}^{NTC_{sk}} TCdifmaxmin(tc_i) \tag{13}$$

3.2.5. Input-Based Test Similarity

The existing literature has shown that a set of diverse test cases has a higher chance to detect faults [45, 46]. For this reason, we have defined two similarity-based objectives. To measure the similarity of test cases we employed the Euclidean distance, which is the similarity measure proposed for simulation models in previous studies [43, 1]. Nevertheless, unlike these previous studies [43, 1], in the context of this study, the TET of test cases might differ, which means that the input and output signals of two test cases can have different numbers of simulation steps. Subsequently, we have adapted the Euclidean distance between test cases to deal with this problem. Given two

signals related to a specific input or output (i.e., sig and sig') in two different test cases, the Euclidean distance is measured following Equation 14. Given two different signals, $\min(k_{sig}, k_{sig'})$ is the number of steps of the signal whose test case has a lower TET, $max_{R_{sig}}$ is the maximum value that the signal in the simulation can obtain, $min_{R_{sig}}$ is the minimum value that the signal in the simulation can obtain and K is the number of steps that the test case with the highest TET in the test suite has. We consider K as the number of steps for the longest test case for two main reasons. First, to ensure that the distances between test cases are normalized. Secondly, because a longer test case might have a higher chance to detect faults, and thus, we penalize the distance of those very short test cases. We remark that in our study the simulation step (Δt) is the same for all test cases. When considering Equation 14, a higher distance means that two signals are more dissimilar.

$$D(sig, sig') = \frac{\sqrt{\sum_{i=0}^{\min(k_{sig}, k_{sig'})} (sig(i \cdot \Delta t) - sig'(i \cdot \Delta t))^2}}{\sqrt{K+1} \times (max_{R_{sig}} - min_{R_{sig}})} \quad (14)$$

Equation 15 defines the input signal-based distance between two different test cases (TC_a and TC_b) for a simulation model with N inputs, which is the sum of the normalized Euclidean distance of each input signal of both test cases. A higher distance means that the test cases are less similar.

$$InTCD(TC_a, TC_b) = \sum_{i=1}^{N} D(Isig_{a_i}, Isig_{b_i}) \quad (15)$$

The input-based test similarity of the overall test suite sk is given in Equation 16, where the distance between all test case pairs in the test suite is considered. A higher TSSimilarity means a higher chance of detecting a fault considering the similarities of test inputs.

$$IN\_TSSim(TS_{sk}) = \sum_{i=1}^{NTC_{sk}} (\sum_{j=i+1}^{NTC_{sk}} InTCD(TC_i, TC_j)) \quad (16)$$

3.2.6. Output-Based Test Similarity

The dissimilarity of a simulation model's outputs can also be considered when selecting test cases. The output-based test similarity is defined like the input-based test similarity as explained in Section 3.2.5. The only difference is that instead of considering the inputs of the Simulation Model for measuring the similarity, the outputs are considered. To this end, the Euclidean distance in Equation 14 is used to measure the distance between an output signal of two test cases. The distance between two test cases is considered by measuring the normalized euclidean distance of each of the output signals in SM. For a given solution sk, the output-based test similarity of the test suite is measured by summing the distance between each test case pair in sk.

3.3. Fitness Function

As mentioned in the introduction, we limited the maximum number of cost-effectiveness measures to three as the NSGA-II suffers from scalability issues with more than three objectives [19]. In all the combinations, TET was the cost objective. By combining the six effectiveness measures, the total number of combinations was n(n-1)/2, being n six (i.e., the number of the six effectiveness measures). The maximum number of combinations among the effectiveness measures was 15. In addition, each of the effectiveness measures was assessed individually with the TET as the cost measures. Thus, by summing up the 15 combinations of effectiveness measures with the six effectiveness measures combined individually with the TET, in total we derived 21 objective combinations and integrated them within NSGA-II. Table 1 summarizes the derived combinations considering up to three cost-effectiveness measures. Regarding the effectiveness objectives, the first derived by selecting one of the effectiveness objectives described in Section 3.2. This resulted in 6 combinations. For the remaining combinations, we employed three objectives, i.e., we combined the six selected effectiveness measures among them and considered TET as the cost measure. As for the crossover and mutation operators, based on a well-known multi-objective test case selection paper [13], we implemented a single point crossover and bit-flip mutation.

4. Experimental Setup

4.1. Research Questions

To evaluate this approach we aimed at answering three Research Questions (RQs) which are detailed as follows:

- ***RQ1 - Sanity Check:*** *How do the selected cost-effectiveness measures together with NSGA-II perform when compared to Random Search (RS)?* This RQ is defined as a sanity check to verify (1) that our test selection problem is non-trivial to solve, and (2) that the selected cost-effectiveness measures are appropriate to select test cases cost-effectively. To address this RQ, we integrated 15 objective combinations with both NSGA-II and RS (which is the baseline algorithm for randomized algorithms [47]) and compared the results.

- ***RQ2 - Comparison with white-box metrics:*** *How do the selected cost-effectiveness measures together with NSGA-II perform when compared to white-box metrics together with NSGA-II?* White-box metrics (such as decision or condition coverage) have been widely employed in regression testing

Table 1: Derived 21 fitness combinations from cost-effectiveness metrics

Combination	Cost	Effect 1	Effect 2
c1	TET	Discontinuity	None
c2			Growth to infinity
c3			Input-based similarity
c4			Instability
c5			MinMax
c6			Output-based similarity
c7		Growth to infinity	None
c8			Input-based similarity
c9			Instability
c10			MinMax
c11			Output-based similarity
c12		Input-based similarity	None
c13			MinMax
c14			Output-based similarity
c15		Instability	None
c16			Input-based similarity
c17			MinMax
c18			Output-based similarity
c19		MinMax	None
c20		Output-based similarity	
c21			MinMax
w1		CC	None
w2		DC	
w3		MC/DC	

[48, 49, 13, 50]. This RQ aims at comparing our black-box regression test selection approach with white-box approaches. To this end, we integrated three white-box metrics within the NSGA-II and compared the results of these approaches with ours.

- ***RQ3 - Combinations Evaluation:*** *Which of the selected cost-effectiveness combinations perform best when solving the black-box test case selection problem?* The second RQ is defined to identify the best objective combination to cost-effectively select test cases within simulation models. To tackle this RQ we compared the performance of the 15 objective combinations explained in Section 3.3 within NSGA-II [35].

- ***RQ4 - Comparison with many-objective algorithms:*** *How do the selected cost-effectiveness measures together with NSGA-II perform when compared to having all of them by employing many-objective algorithms?* It has been shown that NSGA-II is not that good at handling more than three objectives. However, there are other various algorithms that have been shown to be better at handling more than three objectives. Based on [51, 52] we selected two many-objective algorithms (i.e., NSGA-III [40] and MOEA/D [41]) and integrated them with all the proposed cost-effectiveness measures. We figured out that these two algorithms were the most applied for solving many-objective software engineering problems [52]. The objective of this RQ was to compare the selected fitness combinations along with NSGA-II with several objective functional by employing the aforementioned many-objective algorithms.

- ***RQ5 - Improvement Extent:*** *To what extent can NGSA-II together with the best objective combination outperform RS?* The third RQ is defined to identify the percentage of improvement of the best objective combination extracted from RQ2 when compared to RS. To tackle this RQ we extracted the best objective combination from RQ2 and compared it with RS.

4.2. Case studies

Six open source case studies modeled in Simulink of different sizes and complexities were employed to assess our test case selection approach. Table 2 summarizes the key characteristics of these Simulink models, including number of inputs and outputs and number of blocks. The Car Window (CW) case study involves an open-source system consisting of four car windows. The physical layer involves the electrical and mechanical models of the four windows. Each window is controlled by a simulink subsystem that also involves a state machine. The Electro-Mechanical Braking (EMB) system is an industrial open source case study developed by Bosch [53] that combines physical and software models, with the software model controller including a discrete state machine and a continuous PID controller [44]. This case study was used to assess the performance of a test generation algorithm [44]. The Cruise Controller (CC) case study was used in [1] to assess a test case generation approach for Simulink models. This case study only models the controller part of a system and it does not involve any plant model. The Tiny case study was used in [54] for assessing a search-based test data generation approach based on mutation testing and in other studies (e.g., [55]). This is a simple case study involving only 15 blocks. The AC Engine case study involves an AC engine in combination with a controller that involves some safety-critical functionalities. It has been used in previous evaluations by the authors [9, 36, 27]. Lastly, the Two Tanks model involves a two tanks system where a controller regulates the incoming and outgoing flows of the tanks. This case study was used for evaluation of a test oracle generation approach in [56].

For the EMB, CC, Tiny and Two Tanks case studies, 150 test cases were randomly generated. The EMB and the CC case studies were not compatible with Simulink Design Verifier and thus, we could not generate test cases. On the other hand, for the Tiny case study, the number of test cases generated by Simulink Design Verifier was too low, thus, we decided to randomly generate 150 test cases. For the CW case study we employed Simulink Design Verifier to automatically generate test cases following an MC/DC white-box covering approach. We gave the test generator a total of 100,000 seconds to generate test cases. After this time, the test generator yielded a total of 133 test cases. For the AC Engine case study, we reused 120 test cases that were generated for a previous study [9].

4.3. Evaluation Metrics

4.3.1. Modified Hypervolume

To assess the fault revealing capability of the selected test cases using our approach, we employed mutation testing as it has been found to be a good substitute of real faults [57]. Since the selected case studies were modeled in Simulink, we generated mutants considering the mutation operators proposed by Hann et al. [55]. We generated different mutants for each case study and later we executed the generated test cases. After the test case execution we removed some mutants from the initially derived set of mutants. To obtain the final set of mutants we first removed those mutants that were not detected by any of the test cases to avoid the inclusion of equivalent mutants. Secondly, we removed duplicated mutants (i.e., mutants equivalent to one another but not to the original program), as recommended in [58]. Lastly, we removed those mutants that were killable by all test cases. Information related to the number of mutants in both the initial and the final set is available in Table 2. Notice that in Simulink, executing mutants is time consuming and for performance issues we could not generate a large set of mutants. However, our study is in line with other approaches where mutants were used to assess their fault revealing capability within Simulink models (e.g., 13 to 44 mutants [1], 30 mutants [43]).

With the Pareto-frontier returned by the search algorithm, we individually assessed each of the solutions and obtained their mutation score and the test execution time. With this information, we derived a new Pareto-frontier aiming at maximizing the mutation score and minimizing the test execution time. We then employed this Pareto-frontier to obtain the Hypervolume (HV) quality indicator, as recommended by Wang et al. [17]. The HV measures the volume in the objective space by solutions in a Pareto-frontier and it is widely used to assess Pareto-based search algorithms [17, 3, 4]. The higher the HV is, the better the performance of the algorithm. We obtained the HV this way because the ultimate goal of our approach is to detect as many faults as possible reducing the time to execute test cases. The reference points for the HV were, 0% for mutation score and the time to execute the entire test suite (in seconds) for the test execution time.

4.3.2. Average Weighted Sum of Mutation Score and Normalized Test Execution Time

For the first Pareto-frontier (i.e., the one obtained after running the test selection approach), we obtained the mutation score (MS) and normalized test execution time of each of the solutions (W_MS_TET), and computed the weighted sum of both of them (i.e., $W\_MS\_TET = 0.5 \times (1 - norm(TET)) + 0.5 \times MS$). After computing this, we obtained the average value of the W_MS_TET for each of the solutions in the first Pareto-frontier. This permitted us to measure the quality of all the solutions provided by the multi-objective search algorithms along with the proposed fitness combinations without considering decision makers.

The test execution time of a test suite was required to be normalized as this measure could range from hours to days. For this reason, we normalize this metric by measuring the TET of a specific solution (TET_{sk}) with respect to the test suite of the whole initial test suite (TET_{its}) (i.e., $norm(TET_{sk}) = TET_{sk}/TET_{its}$).

4.4. Selected white-box metrics

We selected three white-box metrics: Condition Coverage (CC), Decision Coverage (DC) and Modified Decision/Condition Coverage (MC/DC). There are two main reasons for selecting these three metrics. The first one is that these metrics have been widely investigated and used in the literature [20, 21, 22, 13]. The second one is that our case studies were modeled in MATLAB/Simulink, which provides support for these three white-box metrics. Thus, it was relatively easy to implement a function that could measure the coverage of a test suite in MATLAB.

4.5. Statistical tests

Each algorithm within the selected objective combinations was run 50 times to account for random variations produced by search algorithms, as proposed by Arcuri and Briand [47]. When experimental data was obtained, we applied the Shapiro-Wilk test to assess whether the data was normally distributed. Since the data was not normally distributed, we employed the Mann-Whitney U-test to obtain the statistical significance between two different combinations. We selected a significance level of 5%, meaning that a p-value lower than 0.05 indicated that there was statistical significance. We also employed the Vargha and Delaney $\hat{A}_{12}$ value to assess the difference existing between the techniques.

These statistical tests were employed to answer RQ1, RQ2, RQ3 and RQ4. To keep the paper at a reasonable size, all these statistical tests were included in [59]. As can be appreciated, Tables B6, B7, B9 in [59], which answer to the RQs 1, 2 and RQ4, contain an $\hat{A}_{12}$ value along with

Table 2: Key characteristics of the selected case studies

Case Study	# of Blocks	# of Inputs	# of Outputs	# of Test Cases	Test Generation Criterion	Initial set of mutants	Final set of mutants
CW	235	15	4	133	MC/DC	250	96
EMB	315	1	1	150	Random	40	18
CC	31	5	2	150		60	20
Tiny	15	3	1	150		20	9
AC Engine	257	4	1	120	[27, 36]	20	12
Two Tanks	498	11	7	150	Random	34	6

the p-value. A value higher than 0.5 means that the combination of the black-box metric performed better than the combination in the row (either, being RS, NSGA-II with white-box or one of the two selected many-objective algorithms). A value lower than 0.5 means the opposite. When the value is 0.5, both combinations performed similarly. Additionally, the p-value column indicates the statistical significance between both combinations according to the Mann-Whitney U test (i.e., p-value < 0.05).

Tables B8 and B10 in [59] compared the performance of the 21 different fitness combinations within NSGA-II (i.e., answers to RQ3). Table B8 and B10 show the $\hat{A}_{12}$ values obtained between the selected combinations for the HV quality indicator and the W_MS_TET. A value higher than 0.5 means that the combination in the column performed better than the combination in the row. A value lower than 0.5 means the opposite. When the value is 0.5, both combinations performed similarly. Additionally, the boldface values indicate that there was statistical significance between both combinations according to the Mann-Whitney U test (i.e., p-value < 0.05).

4.6. Algorithm setup

The population size of NSGA-II, NSGA-III and MOEA/D was set to 100 and the total number of fitness evaluations was 25,000. We used a standard single point crossover with a rate of 0.8 and the mutation of a variable was done with the standard probability 1/N, being N the number of variables (i.e., number of test cases). These parameter values were selected based on other studies related to multi-objective test case selection approaches (e.g., [13]), guidelines [47] and other search-based test selection approaches [42, 18, 7]. As for the selection operator, we used the widely used binary tournament selection operator [4, 35]. The number of fitness evaluations for RS were also 25,000.

5. Results and Analysis

This section reports the results obtained from the executed experiments for the two selected evaluation metrics. Table 3 summarizes the results from the statistical tests that can be found in [59]. For each case study and each evaluation metric we provide which is the best obtained black-box fitness combination within NSGA-II. Later, we summarize the results of this best black-box combination for each RQ. For RQ1, the $\hat{A}_{12}$ value obtained when compared to the same combination but with the RS algorithm is provided. For RQ2, we extract which of the three white-box metrics performed best and provide the obtained $\hat{A}_{12}$ value with respect to the best black-box combination. For RQ3, the average $\hat{A}_{12}$ value with respect to the $\hat{A}_{12}$ value of the remaining black-box metrics is provided. For RQ4, we extract the many-objective algorithm that has given the best results and compare the $\hat{A}_{12}$ with the black-box combination along with the NSGA-II. A value higher than 0.5 means that the best black-box combination along with the NSGA-II algorithm performed better than the other algorithm/method being compared. A value lower than 0.5 means the opposite. If the value is in boldface, it means that there was statistical significance according to the Mann-Whitney U-test (i.e., p-value < 0.05). For those of RQ3, we consider that there is statistical significance, if in most of the cases when comparing the best black-box combination there was statistical significance. Although the results are discussed thoroughly in the next sub-sections, it can be appreciated from this table that the proposed method outperformed the remaining for all the cases except for RQ2 within the HV quality indicator. Additionally, despite the proposed method performed better, there was no statistical significance for the W_MS_TET evaluation metric when compared to RS in the AC Engine case study. In addition, the summary distribution of these results can be visualized in Figures 3 and 4.

5.1. Hypervolume quality indicator

In our study, we employed the HV as one of the metrics to measure the effectiveness of our approach. The HV measures the volume in the objective space and it has been widely used to assess Pareto-based search algorithms in the context of search-based software testing [17, 60, 3, 4]. However, it is noteworthy that in our case we do not apply the HV in the traditional way, as explained in Section 4.3.1. The ultimate goal of our approach is to find as many faults as possible in the shortest amount of time. Thus, by considering the mutation score and the test execution time of each solution of the Pareto-frontier, we derived a second Pareto-frontier and obtained the HV. This way we ensured a fair comparison between algorithms and fitness configurations of these algorithms. Notice that the higher the HV, the better the performance of the algorithm.

The first RQ aimed at comparing the NSGA-II algorithm with RS as a sanity check to verify that the problem

Table 3: Summary results of the each RQ with the best BB fitness combination with respect to other RQs

			RQ1		RQ2		RQ3	RQ4	
	Evaluation metric	**Best BB combination**	**Same RS combination**	$\hat{\mathbf{A}}_{12}$	**Best WB combination**	$\hat{\mathbf{A}}_{12}$	**Average** $\hat{\mathbf{A}}_{12}$	**Best Many-objective algorithm**	$\hat{\mathbf{A}}_{12}$
CW	HV	c15	vs RS	**1**	vs DC	**0.77**	**0.9**	vs MOEA/D	**1**
	W_MS_TET	c15	vs RS	**1**	vs DC	**0.22**	**0.98**	vs NSGA-III	**0.94**
EMB	HV	c15	vs RS	**1**	vs MC/DC	**0.97**	**0.798**	vs NSGA-III	**0.92**
	W_MS_TET	c15	vs RS	**1**	vs DC	**0.69**	**0.87**	vs MOEA/D	**1**
CC	HV	c1	vs RS	**1**	vs MC/DC	**0.75**	**0.85**	vs MOEA/D	**1**
	W_MS_TET	c1	vs RS	**1**	vs MC/DC	**0.68**	**0.92**	vs MOEA/D	**1**
Tiny	HV	c15	vs RS	**1**	vs CC	**1**	**0.80**	vs MOEA/D	**1**
	W_MS_TET	c10	vs RS	**1**	vs CC	**0.69**	**0.87**	vs NSGA-III	**0.84**
AC Engine	HV	c15	vs RS	**1**	vs DC	**1**	**0.77**	vs MOEA/D	**1**
	W_MS_TET	c10	vs RS	0.57	vs DC	**1**	**0.79**	vs MOEA/D	**1**
Two Tanks	HV	c3	vs RS	**1**	vs MC/DC	**0**	**0.88**	vs MOEA/D	**1**
	W_MS_TET	c15	vs RS	**1**	vs MC/DC	**0.8**	**0.97**	vs MOEA/D	**1**

to solve was non-trivial. Figure 3 depicts the distribution of the summary of the distribution for the HV indicator for the six case studies.[1] As can be seen, when considering the HV as the quality indicator of this study, the NSGA-II along with the black-box metrics outperformed RS for all cases in the six case studies. These results were further corroborated by means of statistical tests, where NSGA-II outperformed RS with statistical significance according to the Mann-Whitney U-test (i.e., p-value < 0.05), as shown in Table B6 reported in [59].

The second RQ compared the black-box metrics with the white-box ones when selecting test cases of simulation models. From the results (available in [59], and their summary in Figure 3), it can be observed that for the EMB and the Tiny case studies, the distribution of the HV for white-box metrics along with NSGA-II was quite large. In the case of the EMB case study, the distribution in terms of the HV for the white-box approaches was not that large. However, it can be appreciated that the performance of both DC and CC white-box metrics in terms of the HV was quite low, although for the case of MC/DC, the performance was better. The best results for the white-box approaches in terms of the HV indicator were in the Two Tanks case study, which outperformed the black-box fitness combinations with statistical significance. In addition, in terms of the HV, the DC coverage also performed quite well for the CW case study, which outperformed most of the black-box fitness combinations with statistical significance.

When considering the statistical tests for comparing black-box with white-box metrics, for three out of six case studies (EMB, Tiny and AC Engine), all the black-box fitness combinations outperformed those of white-box metrics combinations with statistical significance, as can be seen in Table B7 in [59]. For the CC case study, all the black-box fitness combinations outperformed both CC and DC coverage metrics. However, for this case study, the MC/DC outperformed six black-box metrics fitness combinations with statistical significance, although combinations c1, c15 and c19 outperformed MC/DC with statistical significance. Lastly, for the CW case study, the black-box metrics combinations outperformed both CC and MC/DC white-box coverage approaches with statistical significance. However, CC outperformed 19 out of 21 black-box metrics with statistical significance, although, again, c15 outperformed this white-box coverage metric with statistical significance. Lastly, in the Two Tanks case study, for the HV quality indicator, the white-box metrics performed very well, outperforming the remaining black-box metrics. Nevertheless, the extent when considering average values were not that high (i.e., on average less than 13.5% for the c3 combination.

When considering the HV indicator, for configurations involving only one effectiveness metric and the TET as cost metric (i.e., c1, c7, c12, c15, c19 and c20), we found that overall, c15, which involved the instability effectiveness metric, was the best fitness combination. The only exceptions were c1 for the CC case study and c19 (cruise controller) combination for the CC case study, which slightly outperformed c15, although without statistical significance. In fact, c15 only outperformed c19 with statistical significance in one of the case studies, i.e., CW. Additionally, for the CC case study, c15 also did not outperform with statistical significance combination c7, which included the growth to infinity fitness combination, and in the case of the Two Tanks case study, c15 did not outperform with statistical significance that of c12, which involved the input-based similarity.

When considering all combinations (i.e., c1 to c21), c15 was generally the superior combination. Moreover, c15 outperformed the remaining combinations in two of the case studies (Tiny and CW) with statistical significance, as it can be seen in Table B8 in [59]. It is noteworthy in this case that overall the combinations involving only one effectiveness fitness measure outperformed the combinations involving two effectiveness fitness measures. Conversely, the combination c14 was generally the worst among the selected 21 fitness combinations. This combination was outperformed by most of the remaining combinations with statistical significance. This metric involved both simi-

[1] All the distribution figures are available in [59]

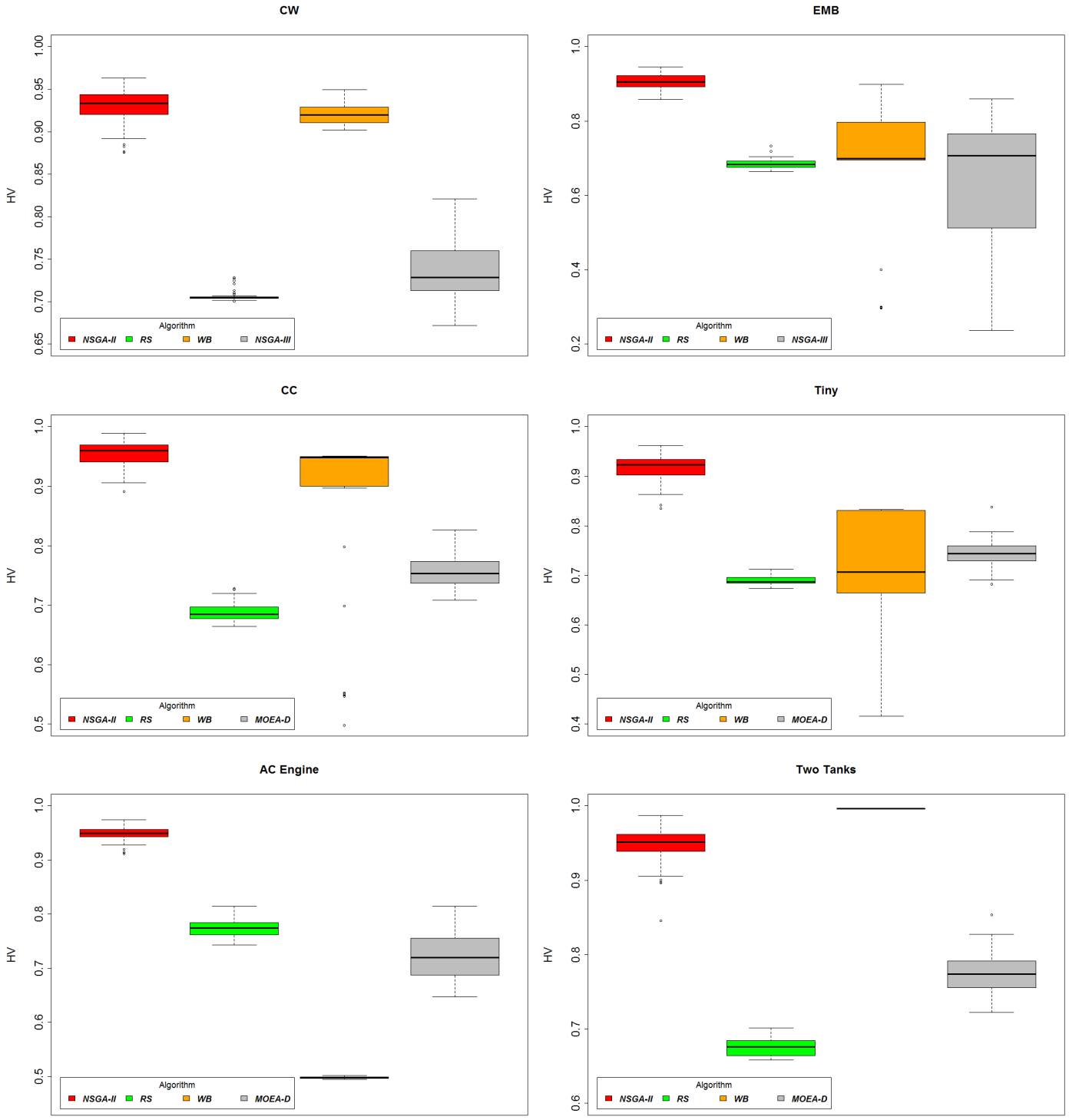

Figure 3: Distribution of the obtained HV quality indicator for the best Black-box metric and with respect to the best algorithms of RQ1, RQ2 and RQ4

larity measures as effectiveness fitness functions (i.e., input and output-based similarity). It is noteworthy, that generally, these metrics did not perform very well for the HV quality indicator. Exceptionally, input-based similarity performed quite well in the Two Tanks case study. In fact, for this case study, c3, which involved the input-based test similarity along with discontinuity performed best.

RQ4 compared the NSGA-II along with the different derived fitness combinations with NSGA-III and MOEA/D, which included all the effectiveness evaluation metrics. The results showed that for the HV quality evaluation, all the derived fitness combinations along with NSGA-II outperformed with statistical significance both NSGA-III and MOEA/D for all the case studies. Overall, MOEA/D per-

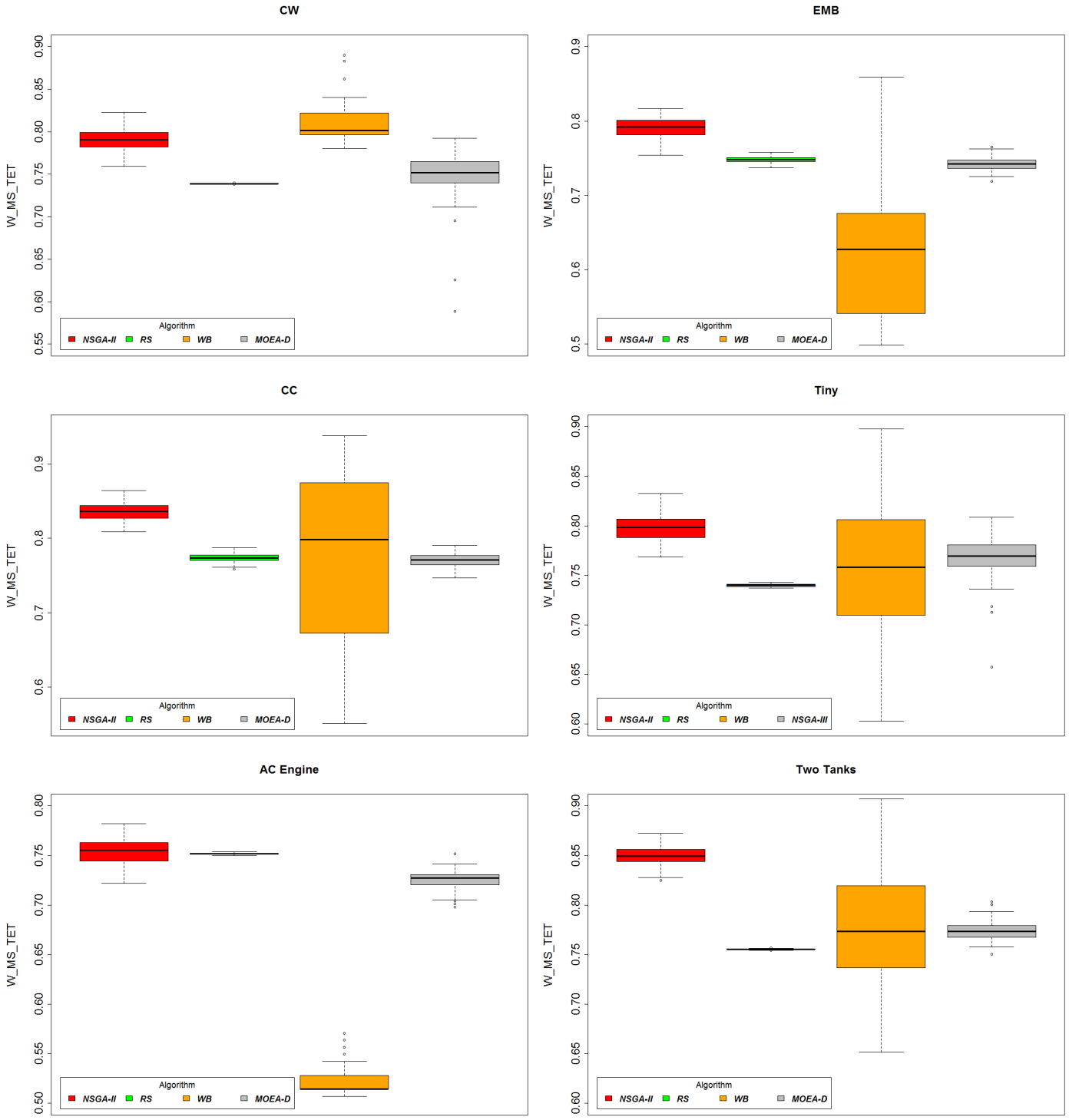

Figure 4: Distribution of the obtained W_MS_TET quality indicator for the best Black-box metric and with respect to the best algorithms of RQ1, RQ2 and RQ4

formed better than NSGA-III, which showed, in general, worse results for the HV evaluation metric. Furthermore, the distribution of the results obtained by NSGA-III is quite large.[2]

To answer RQ5, we extracted for each case study the best combination of NSGA-II and compared the average improvement with respect to RS (Table 4). In four out of six case studies, the combination c15 showed the best performance when considering the HV quality indicator. However, in the CC case study, c1 slightly outperformed c15 when considering average values and c3 in the Two Tanks case study. We compared the average improvement with respect to the same combinations integrated within

[2]Refer to [59] for visualization of results.

Table 4: RQ5: Average improvement of NSGA-II with respect to RS regarding HV metric.

	Best comb.	RS-comb	RS-Best	RS-Avg
CW	c15	24.23 %	23.05 %	24.77 %
EMB	c15	24.47 %	22.38 %	24.03 %
CC	c1	27.96 %	27.30 %	27.94 %
Tiny	c15	25.05 %	23.94 %	25.19 %
ACEngine	c15	18.31 %	14.50 %	18.04 %
Two tanks	c3	28.45 %	27.83 %	28.72 %

RS (column RS-comb in Table 4). We also compared the best combination of NSGA-II with the best combination of RS (Column RS-Best in Table 4). Finally, we compared our approach with respect to an average value of all the combinations of RS (Column RS-Avg in Table 4). As shown in Table 4, for the HV quality indicator, our approach improved RS when configured with the same fitness function on average between 18.31 and 28.45%. When considering the best results for RS, our approach improved it on average between 14.5 and 27.83% for the HV. Lastly, when considering the mean of all the combinations integrated within RS, our approach improved it on average between 18.04 to 28.72%.

5.2. Average Weighted Mutation Score and Normalized Test Execution Time (W_MS_TET)

The second evaluation metric consisted of the average mutation score and test execution time obtained by considering the entire Pareto-frontier. Unlike in the previous metric, in this case we did not filter solutions in a second Pareto-frontier, assuming that in practice, engineers might not have an effective decision maker. It is worth noting the importance of analyzing both results (i.e., mutation score and test suite execution time) together, because a higher mutation score might imply the selection of more test cases and thus, longer test execution times and vice-versa.

As for the first RQ, where NSGA-II is compared with RS, it can be appreciated from the summary boxplots in Figure 4, as well as distribution of all the data available in [59], that generally, NSGA-II outperformed RS in most case studies. This can be further corroborated by means of statistical tests, where NSGA-II outperformed RS with statistical significance in 61 out of 126 cases, whereas RS outperformed NSGA-II with statistical significance in 47 cases. It is noteworthy the AC Engine case study, where RS performed better than NSGA-II for all the cases except the c10. In addition, for the Two Tanks case study RS also outperformed NSGA-II in 11 out of 21 combinations. When not considering these two case studies, the fitness combinations where RS outperformed with statistical significance NSGA-II at least in one case study were c3, c6, c8, c11, c12, c13, c14, c16, c18 and c20. Interestingly, all these fitness combinations included at least one of the similarity measures (i.e., input-based or output based similarity), suggesting that when selecting test cases based on these metrics, an effective decision maker would be required to select the appropriate solution from the Pareto-frontier.

Regarding RQ2, where black-box fitness combinations are compared with white-box fitness combinations, it can be appreciated that, for most case studies, black-box fitness combinations outperformed those of white-box ones for this metric, as appreciated in Figure 4 and [59]. However, when considering statistical tests, there were some exceptions, as shown in Table B.9 in [59]. For the CC case study the MC/DC metric performed quite well, but it only outperformed with statistical significance 6 out of 21 black-box fitness combinations. In the case of the CW case study, most of the white-box metrics outperformed black-box combinations with statistical significance, although only CC outperformed with statistical significance c15. Addditionally, it is noteworthy to mention that for the Two Tanks case study, where in the HV quality evaluation, the three white-box metrics performed very well, for the W_MS_TET evaluation metric, white-box approaches did not perform that well.

When considering RQ3, where different black-box fitness combinations are compared, again for this evaluation metric, c15 was generally the best fitness combination. For this evaluation metric, c15 outperformed the remaining ones in 106 out of 120 cases with statistical significance, as shown in Table B.10 in [59]. The c15 fitness combination was only outperformed with statistical significance once by c1 and c10 only in the CC case study. Conversely, when considering this metric, similar to the HV quality indicator, c14 was the combination showing the worst performance.

As for RQ4, where NSGA-II along with the derived fitness combinations are compared with two many-objective algorithms, within this evaluation metric again, generally, NSGA-II outperformed with statistical significance both algorithms. However, for this evaluation metric there were some exceptions. In fact, NSGA-III performed better than NSGA-II in 11 out of 21 combinations in the Tiny case study and in 17 out of 21 combinations in the CW case study. However, in the remaining case studies, NSGA-II performed better in all case studies, outperforming NSGA-III with statistical significance. As for MOEA/D, it outperformed NSGA-II for 44 out of 126 fitness combinations, although only 30 of them outperformed NSGA-II with statistical significance. Conversely, NSGA-II outperformed MOEA/D in 76 out of 126 fitness combinations with statistical significance. For four case studies and the W_MS_TET evaluation metric, MOEA/D performed better than NSGA-III, whereas in the case study of CW and Tiny, NSGA-III performed better than the MOEA/D.

For answering RQ5 with this metric, similarly as in the HV quality indicator, we extracted for each case study the best combination of NSGA-II and compared the average improvement with respect to RS (Table 5). In three out of six case studies, the combination c15 showed the best performance when considering the W_MS_TET quality indicator (i.e., CW and EMB case studies). Conversely, for the CC case study, c1 obtained the best average vales and for Tiny and the ACEngine case study c10, which

Table 5: RQ5: Average improvement of NSGA-II with respect to RS regarding W_MS_TET metric.

	Best comb.	RS-comb	RS-Best	RS-Avg
CW	c15	6.65 %	4.89 %	5.82 %
EMB	c15	5.27 %	4.79 %	5.95 %
CC	c1	7.42 %	7.16 %	9.03 %
Tiny	c10	7.33 %	5.66 %	7.45 %
ACEngine	c10	0.26 %	-1.49 %	0.08 %
Two tanks	c15	11.08 %	9.32 %	10.78 %

slightly outperformed c15. As in the case of the HV quality indicator, we also compared the best combination of NSGA-II with the same combination but used with RS, the best combination of RS, and average values of all combinations of RS. For this metric, the NSGA-II performed exceptionally worse in the AC Engine case study than in the others. When compared with the same combinations, NSGA-II managed to outperform RS between 0.26 and 11.08% when considering W_MS_TET (Table 5). When compared with the best RS combination, RS managed to slightly outperform NSGA-II, but for the remaining case studies, NSGA-II outperformed RS from 4.79 to 9.32% (Table 5). Lastly, when compared with the average values obtained by RS for the W_MS_TET metric, NSGA-II managed to outperform RS between 0.08 and 10.78%. When not considering the AC case studies, the lowest improvement extents increased to around 5%.

6. Discussion of the Results

We now summarize and discuss the results in relation to the RQs.

RQ1 – Sanity check: The first RQ aims at comparing NSGA-II with RS to assess whether the problem to solve is non-trivial. For the selected evaluation metrics, it can be seen that overall NSGA-II outperformed RS with statistical significance. For the second evaluation metric, i.e., W_MS_TET, RS outperformed in some cases NSGA-II, especially for those combinations in which one of the similarity measures was included as effectiveness measure. Generally, those fitness combinations that included similarity measures did not perform very well. A possible reason for this could be that the Euclidean distance is not an appropriate metric to measure distance among test cases for the case of simulation-based testing. Nevertheless, except for one RS combination in the ACEngine case study, the remaining combinations also outperformed RS when evaluating the approach with this metric. Figures in [59] depict the obtained highest mutation scores for the solutions of the Pareto-frontier. Our approach, for most of the fitness combinations also detects all the mutants, although there are some combinations that at some runs, they have not scored 100% of mutation score (e.g., for some fitness combinations in the Tiny or the CW case studies). Nevertheless, these are only some marginal cases and there is no statistical significance with respect to RS. Notice, however, that in our case, the time it takes to execute the test suite is much lower than RS, and this is what the HV measures, i.e., cost-effectiveness of the entire approach. Thus, we can answer the first RQ as follows:

> *In our experiments, NSGA-II outperformed RS in 5 out of 6 case studies for the two selected evaluation metrics. Based on the experimental results and the statistical tests of our study we can conclude that the test case selection problem for simulation models is a non-trivial problem and thus, multi-objective search algorithms are recommended to be used.*

RQ2 – Comparison with white-box metrics: The second RQ aimed at comparing our approach with white-box metrics as effectiveness fitness functions of the NSGA-II. As discussed in the results section, when considering the HV quality indicator, for most of the case studies our approach outperformed the test case selection method based on white-box metrics with statistical significance. For this evaluation metric, the white-box metrics performed better than black-box metrics for the Two Tanks case study. However, when considering average values, the difference was not very high. In addition, for the white-box metrics performed quite well in the CW case study, especially CC. However, except for the Two Tanks case study, in all the cases there was at least one of the black-box effectiveness combinations that outperformed all the white-box metrics. Our hypothesis behind the white-box metrics performing well in the Two Tanks case study is that the continuous part (i.e., the plant or the environment of the system) of this case study is discretized (i.e., it is modeled with discrete blocks), unlike the remaining case studies (with the exception of Tiny, which is the smallest case study and it does not have continuous subsystems). Our hypothesis matches with the conclusion obtained by Matinnejad et al. [26], where they argue that the original assumption that white-box coverage is a good indicator is not appropriate for all simulation models as it does not account for coverage of the continuous parts of the system.

When considering the second evaluation metric, i.e., W_MS_TET, generally, black-box metrics outperformed white-box ones. For the Two Tanks case study, where with the HV quality indicator, white-box metrics outperformed with statistical significance black-box metrics, all the fitness combinations outperformed with statistical significance CC and DC white-box metrics. MC/DC was the one performing best for this case study, although 10 of the black-box combinations outperformed it with statistical significance.In the CW case study, white-box metrics generally performed quite well, although only CC outperformed c15 with statistical significance. c15 was the black-box combination that generally performed best. When having a closer view to these results, we noticed that when performing the test case selection guiding the search with white-box metrics, the number of selected test cases was quite low. As a result, the TET of the test suites were

quite low and thus, the evaluation metrics were inflated. However, as the white-box metrics-based test selection selected very few test cases, the mutation scores were also much lower than when selecting the test cases by using black-box metrics. In the case of white-box fitness combinations, the highest mutation score of the solutions is lower than all black-box combination metrics for all the cases (except for the Two Tanks case study), as can be seen in the boxplots reported in [59]. This means that despite high test coverage is obtained, there were no solutions in which all the mutants were detected. Additionally, for both metrics, from the boxplots reported in [59] we can find that the variance for black-box techniques was much lower than the variance of white-box metrics. Thus, considering all these facts, we can answer the second RQ as follows:

> The results of white-box metrics were good in some case studies for some of the selected evaluation metrics. These good results were, however, inflated by the low TET of the solutions yielded by the NSGA-II together with traditional white-box effectiveness metrics. However, when considering the mutation score, black-box metrics outperformed white-box ones with statistical significance. Additionally, the variance obtained by black-box techniques is much lower than those obtained by white-box ones. Thus, we believe that for the context of simulation models, selecting test cases guided by black-box metrics is more appropriate than selecting them with white-box metrics.

RQ3 – Combinations Evaluation: The third RQ aimed at obtaining the best combination among the 21 combinations. Overall, c15, which employs the instability anti-pattern as fitness function was the combination performing best for both, the HV and the W_MS_TET metric, although other combinations, (e.g., c1 and c10) also performed quite well when evaluating our approach with the proposed metrics. In fact, c1 outperformed c15 in the CC case study and slightly in the AC Engine case study for the W_MS_TET metric. A possible explanation for this could be that c15 measures the instability degree of a test case, which might be provoked by state-charts when quickly switching between states; however, these two case studies do not contain any state machine, and thus, this anti-pattern might be difficult to be reproduced, unlike the discontinuity anti-pattern, which is the objective function included in c1. Conversely, the combinations performing worse were those that included similarity metrics (i.e., input and output-based similarity). These results were consistent both with our conference version paper [8], as well as with other papers where similarity metrics are compared against anti-patterns for test case generation of simulation models [1, 26]. In fact, the combination c14, which included both metrics as objective functions was the combination that overall performed worst. A possible explanation to this could be that we measured the similarity of test cases with the Euclidean distance, as proposed in [1]. The experiments by these authors also suggested that when using the anti-patterns to guide their search when generating tests, the approach was more effective when detecting faults. Our study also supports their findings but for the problem of test selection. A possible explanation for this could be that the Euclidean distance might not be an effective metric to measure the similarity among test cases within simulation models, and thus, future work can be centered on investigating other metrics. Interestingly, in the case of the car window, all the combinations that included the infinity as one effectiveness measure performed badly. This might be because the outputs of the case study, which included the position of each of the windows in a car, were limited to some maximum and minimum values. As a result, the test cases could not find any situation in the system that provoked a growth to infinity of the output values, resulting in a distortion when employing this anti-pattern for test case selection. This suggests that it is important to analyze the case study before selecting a objective function to guide the search. Overall, we can answer the third research question as follows:

> *The combination c15, which integrates the instability objective function as effectiveness measure, performed the best overall. Other combinations, such as c1 and c10 also performed quite well in some case studies. Conversely, the combinations integrating similarity measures as effectiveness measures do not perform that well. Thus, generally, employing anti-patterns as effectiveness measures to guide the search is more appropriate than employing similarity measures. Particularly, the use of the instability measure is the most appropriate in general.*

RQ4 – Comparison with many-objective algorithms:

RQ4 aimed at answering how the NSGA-II along with the derived fitness combinations performed when compared with many-objective algorithms including all the effectiveness measures along with the TET as cost measure. To this end, based on [51, 52] we selected the NSGA-III and MOEA/D to compare with our approach. Overall, based on the results for the HV quality indicator and the W_MS_TET metric, we found that it was better to select test cases by employing the NSGA-II and one of the selected fitness combinations (refer to the answer of RQ3 to select one of the best combinations). Although there were some cases in which these algorithms outperformed some of the selected fitness combinations along with the NSGA-II, generally, NSGA-II along with one or two effectiveness objectives outperformed with statistical significance the selected many-objective algorithms. Our main theory behind these results is that the effectiveness measures are not generally conflicting among them (even, in some cases,

there might be correlation), and as there are many objective functions within the selected many-objective algorithms, these algorithms discriminate the TET objective, while having optimal results in the most of the remaining objective function. This leads to having long test suites that detect all the mutants but employing a very long TET, what leads to having low HV indicators. Furthermore, another advantage of using less objective functions is that computationally is less expensive, as less objectives need to be measured. Additionally, especially for the NSGA-III, similar to the white-box techniques, the variance of the obtained results was quite large, which is a non-desired effect when deploying a technique in practice. We can thus answer this RQ as follows:

NSGA-II along with one of the derived fitness combinations performed better than NSGA-III and MOEA/D employing all the effectiveness measures.

RQ5 – Improvement extent:

The last RQ aimed at answering to which extent could NSGA-II improve RS. When considering the HV quality indicator, the best black-box combination could outperform RS between around 15 to 28 % on average. When considering the second metric, i.e., the W_MS_TET metric, the improvement extent dropped to around 6 to 11%, with the exception of the AC Engine case study, where, exceptionally, RS slightly performed better than NSGA-II. Overall, these results are positive. When having a closer look into why the improvement extent of W_MS_TET were lower than those measured with the HV, we realized that in the case of the W_MS_TET, there were some solutions with the maximum mutation score, but different TET. When considering the HV, among all these solutions, only the solution with the lowest TET was selected. In practice, this could be controlled by an efficient decision maker, which would select a solution according to the test execution time budget. Nevertheless, even when considering W_MS_TET, the approach proposed in this paper performed better than RS. Thus we can answer the fourth RQ as follows:

The results demonstrated that our approach managed to improve Random Search by between 15 to 28% when considering the HV quality indicator and up to 11% when considering the W_MS_TET metric. The lower improvement extent in W_MS_TET than in HV quality indicator suggests that effective decision-makers could be employed to further improve these results. Notice that in practice, one of the solutions in the Pareto-frontier provided by our approach would need to be chosen, and thus, this deserves to be further investigated in the future.

Concluding Remark – Guidelines:

The use of our proposed cost-effectiveness measures along with NSGA-II significantly outperformed RS and white-box metrics. We recommend the use of the instability measure as an effectiveness measure, or the combination of this measure with one of the other three selected metrics. When information of output signals are not available and tests need to be selected, we recommend the input-based similarity measure, as it is the only measure that does not need a pre-execution of test cases and is more effective than RS.

7. Threats to Validity

We now summarize the identified threats of the performed evaluation, which can be classified in internal validity, conclusion validity, construct validity and external validity.

7.1. Internal validity

Internal factors such as parameters of the algorithm can make internal validity threats arise. A potential internal validity threat of our empirical study is related to the configurations of the search algorithms (e.g., population size), which were not changed. Another configuration might change the performance of the obtained results. Nevertheless, we configured the algorithm considering guidelines related to search algorithms [47] and works including Pareto-based search algorithms for test case selection [13]. Another internal validity threat refers to the generated mutants. In our study we have generated different amounts of mutants depending on the size and complexity of the case study. It should be noted that in our context apart from software, each mutant includes the physical layer of the system, which is typically modeled by complex mathematical equations. This makes the execution of the tests time consuming. This has also happened to us in previous evaluations where Simulink models where employed [61, 7, 9, 62, 63]. However, the amount of mutants is similar to other studies were MATLAB/Simulink models were used as study subjects [1, 61, 7, 9, 43, 64, 65, 66, 67, 55, 26]. Furthermore, we have tried to mitigate this threat by removing duplicated mutants, as recommended by Papadakis et al. [58]. Another internal validity in our study could be how we measure the proposed anti-pattern measures. Within these metrics, in practice it is difficult to differentiate the reason why a test case has certain level of instability or discontinuity (e.g., two test cases can have the same instability level but this might be provoked by different parts of the controller of the system). Such dependency might impact the experimental results, and thus, measuring the independency of test cases could provide better results. To reduce this threat, we combined anti-patterns with similarity metrics to measure independency between test cases. In the future, we plan to investigate the impact of test case dependencies, which was out of the scope of this paper.

7.2. Conclusion validity

A *conclusion validity* threat in our evaluation is related to the random variations. We mitigated this threat by running each algorithm 50 times and applying statistical tests to analyze the results, as recommended by Arcuri and Briand [47].

7.3. Construct validity

Construct validity threats exists in randomized algorithms when the measures used are not comparable across the algorithms. However, to mitigate this threat we used the same stopping criterion for all the algorithms (i.e., we set the total number of fitness evaluations at 25,000).

7.4. External validity

An *external validity* threat in all empirical evaluations in software engineering is related to the generalization of results. We used six case studies, which might not be enough to generalize our findings. However, to reduce this threat we employed simulation models of different characteristics, with different sizes. Furthermore, the EMB case study was an industrial case study developed by Bosch. As for the size of the used case studies, according to a previous paper where 391 public Simulink models were analyzed, more than half of the analyzed models had less than 100 blocks, and around 75% of the case studies had less than 300 blocks [23]. In our case, four of the case studies had more than 200 blocks, being between 235 and 498 blocks, what means that their complexity in terms of size of blocks is higher than most of the public case studies. Furthermore, four of the blocks of the CW case study were state machines with 5 states and 21 transitions each.

8. Related Work

Test case selection has been widely investigated in the current literature. Engströem et al. identified 27 studies encompassing a total of 28 techniques for regression test selection [14], including dataflow-based, modification and changed-based, and coverage-based techniques. Most of these studies are focused on unit testing of software systems. Graves et al. empirically evaluated five regression test selection techniques [20]: safe technique, data-flow technique, minimization technique and three variants of random test selection techniques. Different studies have proposed approaches for regression test selection of different object oriented languages, such as Java [68] or C++ [69]. Safe regression test selection techniques have also been widely investigated in the literature [70, 71]. To solve the test selection problem, other innovative techniques have also been proposed, such as the use of fuzzy-logic [72], or by using semantic differences and similarities between old and new programs to avoid test cases that will produce the same results [73].

Furthermore, in recent years, search-based approaches have gained important attention in the field of test case selection and minimization. Harman argued the need of multi-objective optimization for real-world scenarios [74]. To the best of our knowledge, Pareto-based search algorithms were firstly used for test case selection by Yoo and Harman for testing programs [13]. They formulated the test case selection problem as a multi-objective problem that was instantiated with two versions: (1) a bi-objective formulation that combined coverage and cost and (2) a formulation with three objectives, where, in addition to the coverage and cost, historical information related to faults was included. The same authors later extended this study by proposing an hybrid approach for multi-objective test case selection [75]. Since then, a large corpus of regression test approaches proposed the use of multi-objective search algorithms. Epitropakis et al. empirically evaluated multi-objective algorithms for regression test prioritization [76]. Zheng et al. adapted the MOEA/D for regression test approaches [77]. Maia et al. proposed a multi-objective algorithm considering several aspects related to effectiveness functions (i.e., importance of the modified requirements, dependence among requirements, risk of a test case) as well as several aspects related to cost functions (i.e., test execution time, available test execution time or resources for test execution) [78]. In the context of Software Product Line (SPL) engineering, multi-objective algorithms have been proposed for several problems, including test suite minimization [79, 80] or test case prioritization [32, 37]. Other multi-objective algorithms have combined different sophisticated techniques for regression testing, including fuzzy-logic [81], injecting diversity in genetic algorithms [82], using graphic cards [83] or considering latent semantic indexing [84].

In contrast to most of the test case selection studies, our proposal differs in two main points. The first one is that we only consider information related to the inputs and outputs of the system, unlike most of the studies in the current literature which take advantage of coverage, changes or historical information. The second one is that in our case we aim at selecting test cases to test simulation models, which commonly model systems encompassing software with physical units, in addition to time-continuous sources (e.g., integrators or derivatives). The inputs of these models are signals, which are more complex inputs than those test case inputs for unit testing.

Black-box testing has been widely investigated in software engineering, not only for regression test optimization approaches, but also for test generation [85, 86, 6, 87, 88]. Schroeder and Korel proposed a black-box technique for test suite minimization considering both, inputs and outputs of the program [89]. Since then, black-box metrics have been applied into multiple system applications for test case selection, including database systems [90] or dynamic link library components [91] as well as different code languages (e.g., C [92]). Compared to these studies, the main difference of our approach is the targeted systems, i.e., we use black-box metrics for test selection of simulation models.

Similar to our study, other approaches have proposed cost-effective methods for system level testing without making use of coverage-based information. Pradhan et al. proposed a multi- objective test selection method that can be used when a limited time budget exists [18]. Their fitness function included one cost function (i.e., time difference) and three effectiveness measures (i.e., mean priority, mean probability and mean consequence). Our previous work focused on testing CPS product lines at different test levels and employed several cost-effectiveness measures including historical data or functional and non-functional requirements [7]. Lachmann et al. compared several black box test case selection criteria (e.g., fault revealing history, requirements coverage, etc.) [16]. The main difference between our approach and these approaches is that in our case we do not use any historical data, which is only available when tests have been executed several times. As for similarity-based test case selection, Hemmati et al. proposed a model-based approach [46]. In contrast to our study, their defined similarity measures follow a model-based approach by considering state machines, whereas we consider input and output signals.

Similar to this approach, Matinnejad et al. proposed several anti-patterns as well as similarity based measures for generating test suites to test simulation models with state-charts [43]. In our case, rather than generating test suites we focus on selecting test cases. To this end, we adapted the proposed effectiveness measures in [43] to the test case selection context. Notice however that in the past, the definition of some measures for test suite generation has not prevented successfully re-using the same measures in other verification and validation stages (e.g., in test case prioritization [93]). Besides this, our paper differs from that of [43] in several points. First, our approach is multi-objective, aiming at increasing the effectiveness while reducing costs. In this sense, we evaluated the performance of 21 different objective combinations along with NSGA-II. Conversely, Matinnejad et al. used a single-objective approach to generate test suites using a simulated annealing and an Adaptive Random Testing algorithm [43]. Secondly, when guiding the search based on the anti-patterns, Matinnejad et al. chose one output, whereas in our case, we have normalized each anti-pattern to consider all outputs of a simulation model. Similarly, we adapted the Euclidean distance equation for different TET measures, while in [43] all the test cases have the same TET. Finally, their approach was focused on testing Simulink models that included controllers with state-charts. In our empirical evaluation, we included some models that included state-charts and others that did not include any. Overall, we believe that both approaches are different but complementary.

9. Conclusion and Future work

In this paper we proposed a cost-effective approach to select test cases for simulation models considering only information related to the inputs and outputs of the system. To this end, we have analyzed four different anti-pattern and two similarity measures and proposed different measures to cost-effectively select test cases. We formed in total 21 combinations and integrated them within NSGA-II. An empirical evaluation with six different case studies showed that our approach is cost-effective when compared to RS, improving them in about 18 to 28% when considering the HV quality indicator. Moreover, for the second evaluation metric (i.e., W_MS_TET), NSGA-II outperformed RS in 5 out of 6 case studies. Furthermore, the proposed approach outperformed NSGA-II along with three traditional white-box metrics (i.e., CC, DC and MC/DC coverage) for most of the case studies, as well as two many-objective algorithms (i.e., NSGA-III and MOEA/D). When considering the proposed combinations, the measures based on anti-patterns performed better than both measures based on similarities. Particularly, the instability measure generally outperformed the remaining ones.

In the future, we would like to investigate other distance metrics to measure the similarity among test cases. To this end, we foresee adapting traditional similarity metrics (e.g., Hamming distance, Jaccard, etc.) to the context of simulation-based testing. This could enable us to find why similarity measures did not perform well in this context. Furthermore, to deploy this technique in practice, decision-makers would need to be developed and investigated, as one solution from the Pareto-frontier returned by the NSGA-II would need to be selected. Future direction will aim to develop, investigate and evaluate decision makers for application in practice.

Material

For the sake of replicability, experimental material is available in `https://github.com/aitorarrietamarcos/IST2019Paper`.

Acknowledgments

The work has been developed by the embedded systems group supported by the Department of Education, Universities and Research of the Basque Government.

References

[1] R. Matinnejad, S. Nejati, L. C. Briand, T. Bruckmann, Automated test suite generation for time-continuous simulink models, in: Proceedings of the 38th International Conference on Software Engineering, ICSE '16, ACM, New York, NY, USA, 2016, pp. 595–606.

[2] L. Briand, S. Nejati, M. Sabetzadeh, D. Bianculli, Testing the untestable: Model testing of complex software-intensive systems, in: Proceedings of the 38th International Conference on Software Engineering Companion, ICSE '16, ACM, 2016, pp. 789–792. doi:10.1145/2889160.2889212.

[3] R. Ben Abdessalem, S. Nejati, L. C. Briand, T. Stifter, Testing advanced driver assistance systems using multi-objective search and neural networks, in: Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, 2016, pp. 63–74. doi:10.1145/2970276.2970311.

[4] R. Ben Abdessalem, S. Nejati, L. C. Briand, T. Stifter, Testing vision-based control systems using learnable evolutionary algorithms, in: Proceedings of the 40th International Conference on Software Engineering, ICSE '18, 2018.

[5] G. Sagardui, L. Etxeberria, J. A. Agirre, A. Arrieta, C. F. Nicolas, J. M. Martin, A configurable validation environment for refactored embedded software: An application to the vertical transport domain, in: 2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), IEEE, 2017, pp. 16–19.

[6] A. Arcuri, M. Z. Iqbal, L. Briand, Black-box system testing of real-time embedded systems using random and search-based testing, in: Proceedings of the 22Nd IFIP WG 6.1 International Conference on Testing Software and Systems, ICTSS'10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 95–110.

[7] A. Arrieta, S. Wang, G. Sagardui, L. Etxeberria, Search-based test case selection of cyber-physical system product lines for simulation-based validation, in: Proceedings of the 20th International Systems and Software Product Line Conference, 2016, pp. 297–306.

[8] A. Arrieta, S. Wang, A. Arruabarrena, U. Markiegi, G. Sagardui, L. Etxeberria, Multi-objective black-box test case selection for cost-effectively testing simulation models, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '18, ACM, New York, NY, USA, 2018, pp. 1411–1418. doi:10.1145/3205455.3205490.
URL http://doi.acm.org/10.1145/3205455.3205490

[9] A. Arrieta, S. Wang, G. Sagardui, L. Etxeberria, Search-based test case prioritization for simulation-based testing of cyber-physical system product lines, Journal of Systems and Software 149 (2019) 1 – 34. doi:https://doi.org/10.1016/j.jss.2018.09.055.
URL http://www.sciencedirect.com/science/article/pii/S0164121218302085

[10] A. Kane, T. E. Fuhrman, P. Koopman, Monitor based oracles for cyber-physical system testing: Practical experience report, in: 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2014, Atlanta, GA, USA, June 23-26, 2014, 2014, pp. 148–155.

[11] G. Sagardui, J. Agirre, U. Markiegi, A. Arrieta, C. F. Nicolás, J. M. Martín, Multiplex: A co-simulation architecture for elevators validation, in: Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM), 2017 IEEE International Workshop of, IEEE, 2017, pp. 1–6.

[12] C. A. González, M. Varmazyar, S. Nejati, L. C. Briand, Y. Isasi, Enabling model testing of cyber-physical systems, in: Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS '18, ACM, New York, NY, USA, 2018, pp. 176–186. doi:10.1145/3239372.3239409.
URL http://doi.acm.org/10.1145/3239372.3239409

[13] S. Yoo, M. Harman, Pareto efficient multi-objective test case selection, in: Proceedings of the 2007 international symposium on Software testing and analysis, ACM, 2007, pp. 140–150.

[14] E. Engström, P. Runeson, M. Skoglund, A systematic review on regression test selection techniques, Information and Software Technology 52 (1) (2010) 14–30.

[15] Fmi standard.

[16] R. Lachmann, M. Felderer, M. Nieke, S. Schulze, C. Seidl, I. Schaefer, Multi-objective black-box test case selection for system testing, in: Proceedings of the Genetic and Evolutionary Computation Conference, ACM, 2017, pp. 1311–1318.

[17] S. Wang, S. Ali, T. Yue, Y. Li, M. Liaaen, A practical guide to select quality indicators for assessing pareto-based search algorithms in search-based software engineering, in: Proceedings of the 38th International Conference on Software Engineering, ICSE '16, 2016, pp. 631–642. doi:10.1145/2884781.2884880.

[18] D. Pradhan, S. Wang, S. Ali, T. Yue, Search-based cost-effective test case selection within a time budget: An empirical study, in: Proceedings of the Genetic and Evolutionary Computation Conference 2016, GECCO '16, ACM, New York, NY, USA, 2016, pp. 1085–1092. doi:10.1145/2908812.2908850.
URL http://doi.acm.org/10.1145/2908812.2908850

[19] A. Panichella, F. Kifetew, P. Tonella, Automated test case generation as a many-objective optimisation problem with dynamic selection of the targets, IEEE Transactions on Software Engineering.

[20] T. L. Graves, M. J. Harrold, J.-M. Kim, A. Porter, G. Rothermel, An empirical study of regression test selection techniques, ACM Transactions on Software Engineering and Methodology (TOSEM) 10 (2) (2001) 184–208.

[21] S. Rayadurgam, M. P. E. Heimdahl, Coverage based test-case generation using model checkers, in: Engineering of Computer Based Systems, 2001. ECBS 2001. Proceedings. Eighth Annual IEEE International Conference and Workshop on the, IEEE, 2001, pp. 83–91.

[22] J. A. Jones, M. J. Harrold, Test-suite reduction and prioritization for modified condition/decision coverage, IEEE Transactions on software Engineering 29 (3) (2003) 195–209.

[23] S. A. Chowdhury, S. Mohian, S. Mehra, S. Gawsane, T. T. Johnson, C. Csallner, Automatically finding bugs in a commercial cyber-physical system development tool chain with slforge, in: Proceedings of the 40th International Conference on Software Engineering, ACM, 2018, pp. 981–992.

[24] H. Shokry, M. Hinchey, Model-based verification of embedded software, Computer 42 (4) (2009) 53 – 59.

[25] R. Matinnejad, S. Nejati, L. Briand, T. Bruckmann, C. Poull, Search-based automated testing of continuous controllers: Framework, tool support, and case studies, Information and Software Technology 57 (2015) 705 – 722.

[26] R. Matinnejad, S. Nejati, L. Briand, T. Bruckmann, Test generation and test prioritization for simulink models with dynamic behavior, IEEE Transactions on Software Engineering.

[27] A. Arrieta, S. Wang, U. Markiegi, G. Sagardui, L. Etxeberria, Search-based test case generation for cyber-physical systems, in: Evolutionary Computation (CEC), 2017 IEEE Congress on, 2017, pp. 688–697.

[28] A. Arrieta, G. Sagardui, L. Etxeberria, J. Zander, Automatic generation of test system instances for configurable cyber-physical systems, Software Quality Journal 25 (3) (2017) 1041–1083.

[29] J. C. Eidson, E. A. Lee, S. Matic, Distributed real-time software for cyber-physical systems, Proceedings of the IEEE 100 (2011) 45–59.

[30] M. Harman, Making the case for morto: Multi objective regression test optimization, in: Software Testing, Verification and Validation Workshops (ICSTW), 2011 IEEE Fourth International Conference on, IEEE, 2011, pp. 111–114.

[31] R. M. Hierons, M. Li, X. Liu, S. Segura, W. Zheng, Sip: Optimal product selection from feature models using many-objective evolutionary optimization, ACM Transactions on Software Engineering and Methodology (TOSEM) 25 (2) (2016) 17.

[32] S. Wang, D. Buchmann, S. Ali, A. Gotlieb, D. Pradhan, M. Liaaen, Multi-objective test prioritization in software product line testing: An industrial case study, in: Proceedings of the 18th International Software Product Line Conference - Volume 1, SPLC '14, ACM, New York, NY, USA, 2014, pp. 32–41. doi:10.1145/2648511.2648515.

[33] F. Sarro, A. Petrozziello, M. Harman, Multi-objective software effort estimation, in: Proceedings of the 38th International Conference on Software Engineering, ACM, 2016, pp. 619–630.

[34] J. Campos, Y. Ge, G. Fraser, M. Eler, A. Arcuri, An empirical evaluation of evolutionary algorithms for test suite generation, in: Symposium on Search-Based Software Engineering, 2017.

[35] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii, IEEE transactions on evolutionary computation 6 (2) (2002) 182–197.

[36] A. Arrieta, S. Wang, U. Markiegi, G. Sagardui, L. Etxeber-

ria, Employing multi-objective search to enhance reactive test case generation and prioritization for testing industrial cyber-physical systems, IEEE Transactions on Industrial Informatics 14 (3) (2018) 1055–1066.

[37] J. A. Parejo, A. B. Sánchez, S. Segura, A. Ruiz-Cortés, R. E. Lopez-Herrejon, A. Egyed, Multi-objective test case prioritization in highly configurable systems: A case study, Journal of Systems and Software (2016) –.

[38] E. Zitzler, M. Laumanns, L. Thiele, et al., Spea2: Improving the strength pareto evolutionary algorithm, in: Eurogen, Vol. 3242, 2001, pp. 95–100.

[39] D. W. Corne, N. R. Jerram, J. D. Knowles, M. J. Oates, Pesa-ii: Region-based selection in evolutionary multiobjective optimization, in: Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, Morgan Kaufmann Publishers Inc., 2001, pp. 283–290.

[40] K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints., IEEE Trans. Evolutionary Computation 18 (4) (2014) 577–601.

[41] Q. Zhang, H. Li, Moea/d: A multiobjective evolutionary algorithm based on decomposition, IEEE Transactions on evolutionary computation 11 (6) (2007) 712–731.

[42] S. Wang, S. Ali, A. Gotlieb, Minimizing test suites in software product lines using weight-based genetic algorithms, in: Proceedings of the 2013 Genetic and Evolutionary Computation Conference, Amsterdam, Netherlands, 2013, pp. 1493 – 1500.

[43] R. Matinnejad, S. Nejati, L. C. Briand, T. Bruckmann, Effective test suites for mixed discrete-continuous stateflow controllers, in: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ACM, 2015, pp. 84–95.

[44] R. Matinnejad, S. Nejati, L. C. Briand, Automated testing of hybrid simulink/stateflow controllers: Industrial case studies, in: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, ACM, New York, NY, USA, 2017, pp. 938–943. doi:10.1145/3106237.3117770.
URL http://doi.acm.org/10.1145/3106237.3117770

[45] R. Feldt, S. M. Poulding, D. Clark, S. Yoo, Test set diameter: Quantifying the diversity of sets of test cases, in: 2016 IEEE International Conference on Software Testing, Verification and Validation, ICST 2016, Chicago, IL, USA, April 11-15, 2016, 2016, pp. 223–233. doi:10.1109/ICST.2016.33.

[46] H. Hemmati, A. Arcuri, L. Briand, Achieving scalable model-based testing through test case diversity, ACM Transactions on Software Engineering and Methodology 22 (1) (2013) 6:1–6:42.

[47] A. Arcuri, L. Briand, A practical guide for using statistical tests to assess randomized algorithms in software engineering, in: Software Engineering (ICSE), 2011 33rd International Conference on, IEEE, 2011, pp. 1–10.

[48] S. Elbaum, A. Malishevsky, 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, 2001, pp. 329–338.

[49] S. Elbaum, G. Rothermel, J. Penix, Techniques for improving regression testing in continuous integration development environments, in: Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE'14), ACM, 2014, pp. 235–245.

[50] S. Yoo, M. Harman, Regression testing minimization, selection and prioritization: a survey, Software Testing, Verification and Reliability 22 (2) (2012) 67–120.

[51] B. Li, J. Li, K. Tang, X. Yao, Many-objective evolutionary algorithms: A survey, ACM Computing Surveys (CSUR) 48 (1) (2015) 13.

[52] A. Ramírez, J. R. Romero, S. Ventura, A survey of many-objective optimisation in search-based software engineering, Journal of Systems and Software.

[53] T. Strathmann, J. Oehlerking., Verifying properties of an electro-mechanical braking system, in: In 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems, 2015, pp. 49–56.

[54] Y. Zhan, J. A. Clark, A search-based framework for automatic testing of matlab/simulink models, Journal of Systems and Software 81 (2) (2008) 262–285.

[55] L. T. M. Hanh, N. T. Binh, K. T. Tung, A novel fitness function of metaheuristic algorithms for test data generation for simulink models based on mutation analysis, Journal of Systems and Software 120 (C) (2016) 17–30.

[56] C. Menghi, S. Nejati, K. Gaaloul, L. Briand, Generating automated and online test oracles for simulink models with continuous and uncertain behaviors, arXiv preprint arXiv:1903.03399.

[57] R. Just, D. Jalali, L. Inozemtseva, M. D. Ernst, R. Holmes, G. Fraser, Are mutants a valid substitute for real faults in software testing?, in: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, ACM, 2014, pp. 654–665.

[58] M. Papadakis, Y. Jia, M. Harman, Y. Le Traon, Trivial compiler equivalence: A large scale empirical study of a simple, fast and effective equivalent mutant detection technique, in: Proceedings of the 37th International Conference on Software Engineering-Volume 1, IEEE Press, 2015, pp. 936–946.

[59] A. Arrieta, Statistical tests of the results (2019).
URL https://sites.google.com/mondragon.edu/sse-ist-2019

[60] S. Wang, S. Ali, T. Yue, Ø. Bakkeli, M. Liaaen, Enhancing test case prioritization in an industrial setting with resource awareness and multi-objective search, in: Software Engineering Companion (ICSE-C), IEEE/ACM International Conference on, IEEE, 2016, pp. 182–191.

[61] A. Arrieta, S. Wang, G. Sagardui, L. Etxeberria, Test case prioritization of configurable cyber-physical systems with weight-based search algorithms, in: Proceedings of the Genetic and Evolutionary Computation Conference 2016, GECCO '16, ACM, New York, NY, USA, 2016, pp. 1053–1060. doi:10.1145/2908812.2908871.

[62] U. Markiegi, A. Arrieta, G. Sagardui, L. Etxeberria, Search-based product line fault detection allocating test cases iteratively, in: Proceedings of the 21st International Systems and Software Product Line Conference - Volume A, SPLC '17, ACM, New York, NY, USA, 2017, pp. 123–132. doi:10.1145/3106195.3106210.
URL http://doi.acm.org/10.1145/3106195.3106210

[63] A. Arrieta, S. Segura, U. Markiegi, G. Sagardui, L. Etxeberria, Spectrum-based fault localization in software product lines, Information and Software Technology 100 (2018) 18–31.

[64] B. Liu, L. Lucia, S. Nejati, L. Briand, Improving fault localization for simulink models using search-based testing and prediction models, in: 24th IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER 2017), 2017.

[65] B. Liu, S. Nejati, L. C. Briand, et al., Effective fault localization of automotive simulink models: achieving the trade-off between test oracle effort and fault localization accuracy, Empirical Software Engineering (2018) 1–47.

[66] B. Liu, Lucia, S. Nejati, L. C. Briand, T. Bruckmann, Simulink fault localization: an iterative statistical debugging approach, Software Testing, Verification and Reliability 26 (6) (2016) 431–459.

[67] K. T. Le Thi My Hanh, N. T. B. Tung, Mutation-based test data generation for simulink models using genetic algorithm and simulated annealing, International Journal of Computer and Information Technology 3 (04) (2014) 763–771.

[68] M. J. Harrold, J. A. Jones, T. Li, D. Liang, A. Orso, M. Pennings, S. Sinha, S. A. Spoon, A. Gujarathi, Regression test selection for java software, in: ACM Sigplan Notices, Vol. 36, ACM, 2001, pp. 312–326.

[69] G. Rothermel, M. J. Harrold, J. Dedhia, Regression test selection for c++ software, Software Testing, Verification and Reliability 10 (2) (2000) 77–109.

[70] G. Rothermel, M. J. Harrold, A safe, efficient regression test selection technique, ACM Transactions on Software Engineering and Methodology (TOSEM) 6 (2) (1997) 173–210.

[71] G. Rothermel, M. J. Harrold, Empirical studies of a safe regression test selection technique, CSE Journal Articles (1998) 11.
[72] Z. Xu, K. Gao, T. M. Khoshgoftaar, Application of fuzzy expert system in test case selection for system regression test, in: Information Reuse and Integration, Conf, 2005. IRI-2005 IEEE International Conference on., IEEE, 2005, pp. 120–125.
[73] D. Binkley, Reducing the cost of regression testing by semantics guided test case selection, in: Software Maintenance, 1995. Proceedings., International Conference on, IEEE, 1995, pp. 251–260.
[74] M. Harman, Making the case for morto: Multi objective regression test optimization, in: 2011 Fourth International Conference on Software Testing, Verification and Validation Workshops, IEEE, 2011, pp. 111–114.
[75] S. Yoo, M. Harman, Using hybrid algorithm for pareto efficient multi-objective test suite minimisation, Journal of Systems and Software 83 (4) (2010) 689–701.
[76] M. G. Epitropakis, S. Yoo, M. Harman, E. K. Burke, Empirical evaluation of pareto efficient multi-objective regression test case prioritisation, in: Proceedings of the 2015 International Symposium on Software Testing and Analysis, ISSTA 2015, ACM, New York, NY, USA, 2015, pp. 234–245.
[77] W. Zheng, R. M. Hierons, M. Li, X. Liu, V. Vinciotti, Multi-objective optimisation for regression testing, Information Sciences 334 (2016) 1–16.
[78] C. L. B. Maia, R. A. F. do Carmo, F. G. de Freitas, G. A. L. de Campos, J. T. de Souza, A multi-objective approach for the regression test case selection problem, in: Proceedings of Anais do XLI Simposio Brasileiro de Pesquisa Operacional (SBPO 2009), 2009, pp. 1824–1835.
[79] S. Wang, S. Ali, A. Gotlieb, Cost-effective test suite minimization in product lines using search techniques, Journal of Systems and Software 103 (0) (2015) 370 – 391.
[80] H. Baller, S. Lity, M. Lochau, I. Schaefer, Multi-objective test suite optimization for incremental product family testing, in: Software Testing, Verification and Validation (ICST), 2014 IEEE Seventh International Conference on, IEEE, 2014, pp. 303–312.
[81] Z. Anwar, A. Ahsan, Multi-objective regression test suite optimization with fuzzy logic, in: Multi Topic Conference (INMIC), 2013 16th International, IEEE, 2013, pp. 95–100.
[82] A. Panichella, R. Oliveto, M. Di Penta, A. De Lucia, Improving multi-objective test case selection by injecting diversity in genetic algorithms, IEEE Transactions on Software Engineering 41 (4) (2015) 358–383.
[83] S. Yoo, M. Harman, S. Ur, Highly scalable multi objective test suite minimisation using graphics cards, in: International Symposium on Search Based Software Engineering, Springer, 2011, pp. 219–236.
[84] M. M. Islam, A. Marchetto, A. Susi, G. Scanniello, A multi-objective technique to prioritize test cases based on latent semantic indexing, in: 2012 16th European Conference on Software Maintenance and Reengineering, IEEE, 2012, pp. 21–30.
[85] T. Y. Chen, H. Leung, I. Mak, Adaptive random testing., in: ASIAN, Vol. 4, Springer, 2004, pp. 320–329.
[86] T. Y. Chen, F.-C. Kuo, R. G. Merkel, S. P. Ng, Mirror adaptive random testing, Information and Software Technology 46 (15) (2004) 1001–1010.
[87] T. Y. Chen, F.-C. Kuo, R. G. Merkel, T. Tse, Adaptive random testing: The art of test case diversity, Journal of Systems and Software 83 (1) (2010) 60–66.
[88] I. Ciupa, A. Leitner, M. Oriol, B. Meyer, Artoo: adaptive random testing for object-oriented software, in: Proceedings of the 30th international conference on Software engineering, ACM, 2008, pp. 71–80.
[89] P. J. Schroeder, B. Korel, Black-box test reduction using input-output analysis, in: Proceedings of the 2000 ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA '00, ACM, New York, NY, USA, 2000, pp. 173–177. doi:10.1145/347324.349042.
URL http://doi.acm.org/10.1145/347324.349042
[90] E. Rogstad, L. Briand, R. Torkar, Test case selection for black-box regression testing of database applications, Information and Software Technology 55 (10) (2013) 1781–1795.
[91] J. Zheng, L. Williams, B. Robinson, K. Smiley, Regression test selection for black-box dynamic link library components, in: Proceedings of the Second International Workshop on Incorporating COTS Software into Software Systems: Tools and Techniques, IEEE Computer Society, 2007, p. 9.
[92] T. Xie, D. Notkin, Checking inside the black box: Regression testing by comparing value spectra, IEEE Transactions on software Engineering 31 (10) (2005) 869–883.
[93] C. Henard, M. Papadakis, M. Harman, Y. Jia, Y. Le Traon, Comparing white-box and black-box test prioritization, in: Proceedings of the 38th International Conference on Software Engineering, ACM, 2016, pp. 523–534.

Appendix A. Distribution of the obtained results

The distribution of the obtained results can be found by accessing the following URL: `https://sites.google.com/mondragon.edu/sse-ist-2019/Distribution-of-the-obtained-results-21?authuser=0`

Appendix B. Results for statistical tests

The summary of the statistical tests can be found by accessing the following URL: `https://sites.google.com/mondragon.edu/sse-ist-2019/Results-for-statistical-tests?authuser=0`