Employing Multi-Objective Search to Enhance Reactive Test Case Generation and Prioritization for Testing Industrial Cyber-Physical Systems

Aitor Arrieta, Shuai Wang, Urtzi Markiegi, Goiuria Sagardui, and Leire Etxeberria

***Abstract*—The test case generation and prioritization of industrial cyber-physical systems face critical challenges, and simulation-based testing is one of the most commonly used techniques for testing these complex systems. However, simulation models of industrial CPSs are usually very complex, and executing the simulations becomes computationally expensive, which often make it infeasible to execute all the test cases. To address these challenges, this paper proposes a multi-objective test generation and prioritization approach for testing industrial CPSs by defining a fitness function with four objectives and designing different crossover and mutation operators. We empirically evaluated our fitness function and designed operators along with five multi-objective search algorithms [e.g., nondominated sorting genetic algorithm (NSGA-II)] using four case studies. The evaluation results demonstrated that NSGA-II achieved significantly better performance than the other algorithms and managed to improve random search for on average 43.80% for each objective and 49.25% for the quality indicator hypervolume.**

***Index Terms*—Automated validation, cyber-physical systems, test generation.**

I. Introduction

CYBER-PHYSICAL systems (CPSs), which integrate digital cyber technologies (such as microprocessors or complex networks) with parallel physical processes [1], are attracting increasing attention from both academics and industry. Typically, microprocessors monitor and alter the physical processes by means of sensors and actuators (e.g., electrical engines). Examples of CPSs include smart pick-and-place machines [2], pacemakers [3], and unmanned aerial vehicles (UAVs) [4]. CPSs are starting to have a high impact on industrial automation for different applications such as manufacturing, electronics assembly, continuous processing, or energy management [5]. Complex industrial CPSs often include heterogeneous components such as electromechanical, thermal, computing, and communication elements, interconnected among them and encompassing environmental effects [6].

CPSs have been cataloged as untestable systems, and traditional testing techniques [e.g., model-based testing (MBT)] are usually expensive, time-consuming, or infeasible to apply [7]. This is due to the fact that it is challenging for the traditional testing techniques to capture complex continuous dynamics and interactions between the system and its environment (e.g., people walking around when automatic braking systems are in use for automotive systems) [7], [8]. As a result, simulation-based testing has been envisioned as an efficient means to test CPSs in a systematic and automated manner [7]. The use of simulation models permits several advantages such as

1) the execution of larger test cases (TCs),
2) selection of critical scenarios,
3) specification of test oracles for the automated validation of the system, or
4) replication of safety-critical functions of a real system where it is expensive to execute TCs [7], [9].

Therefore, testing CPSs face different particularities (e.g., concurrency or timing behavior) as compared to testing traditional software.

Aside from this, CPSs frequently interact with their environment, which makes testing CPSs unpredictable [9]. To deal with this problem, different approaches have focused on testing CPSs using reactive TCs [10] since reactive TCs are able to observe different physical properties of the environment at runtime. Moreover, reactive TCs are composed of a set of signals that stimulate the inputs of CPSs under test, and a set of properties (e.g., time, system outputs) is monitored to react on them and change the stimulation signal values [10]. Furthermore, the input space of a CPS is usually extremely large [8], with these inputs associated with requirements, which again bring great challenges for thoroughly testing a given CPS. In addition, diversification of the input space has helped the detection of faults in the past [11]. The time to execute TCs for testing CPSs is usually long (from some seconds to some minutes [12]). This is due to several reasons:

Manuscript received July 7, 2017; revised October 16, 2017 and November 20, 2017; accepted December 6, 2017. Date of publication December 29, 2017; date of current version March 1, 2018. This work was developed by the Embedded Systems Group supported by the Department of Education, Universities and Research of the Basque Government. The work of S. Wang was supported in part by the Research Council of Norway funded Certus SFI, in part by RFF Hovedstaden funded MBE-CR project, and in part by COST Action CA15140 (ImAppNIO). Paper no. TII-17-1478. *(Corresponding author: Aitor Arrieta.)*

A. Arrieta, U. Markiegi, G. Sagardui, and L. Etxeberria are with the Electronics and Computing Department, University of Mondragon, Mondragon 20500, Spain (e-mail: aarrieta@mondragon.edu; umarkiegi@mondragon.edu; gsagardui@mondragon.edu; letxeberria@mondragon.edu).

S. Wang is with the Certus V&V Center, Simula Research Laboratory, Fornebu 1364, Norway (e-mail: shuai@simula.no).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2017.2788019

1) When testing a CPS under simulation, the physical layer has to be considered, which is often modeled with complex mathematical models;
2) CPSs often need to be tested under different test levels [e.g., model-in-the-loop (MiL), software-in-the-loop (SiL), and hardware-in-the-loop (HiL)]. For instance, the HiL test level employs a real-time infrastructure, and thus, the simulation is in real time, which is not the case for the traditional software testing; and
3) CPSs may involve a large number of configurations with various parameters (e.g., different controllers parameters), and thus, it may take a huge amount of time to test different valid configurations.

Furthermore, there are several factors that make the generation of reactive TCs for CPSs a challenge. More specifically, reactive TCs usually have many inputs, e.g., different number of states and stimuli signals with many potential values. For instance, for the UAV case study (obtained from our industrial partner), one state can have ten stimuli signals (six Booleans and four integers). The four integers (I) have the following maximum and minimum values, I1 = [0, 360], I2 = [0, 500], I3 = [−2000, 2000], and I4 = [−2000, 2000]. Thus, the search space for a single state of a reactive TC is $2^6 \times 361 \times 501 \times 4001 \times 4001 = 1.85\text{E} + 14$. When there is a need to generate TCs with a number of states, the search space would grow exponentially. Thus, it is practically infeasible to evaluate all the combinations of states/signals with all possible values, i.e., the search space in this context is too huge to be fully explored. Moreover, multi-objective search-based approaches based on evolutionary algorithms are particularly appropriate to be applied when there is a need to balance more than one conflicting objectives, which suits to address our TC generation problem. For instance, a higher requirements coverage (RC) usually indicates more number of TCs to be executed with longer test execution time (TET). Apart from TC generation, TC prioritization in CPSs is highly important to detect faults as fast as possible, or, when employing reactive TCs, to reduce the TET. However, this prioritization is not trivial due to several reasons [10]. First, when testing CPSs at a system level, the TET of each TC is in the order of seconds to minutes. Second, the test suite (TS) for testing CPSs is typically composed of many TCs, what makes the search space for the prioritization extremely large [10]. Finally, when prioritizing TCs, a historical database incorporating attributes of TCs (e.g., code coverage) is typically required, which is not always available.

Note that we have been conducting industrial-oriented research by collaborating with three industrial partners for testing industrial CPSs, which include Orona (elevators company), Ulma Handling (automated warehouses), and Alerion (UAVs). Based on our collaboration with these partners and their needs, we observed the following desirable characteristics when generating TCs for testing CPSs.

1) traceability between TCs and requirements, being possible to cover all the requirements,
2) effective TCs for finding faults, and
3) low test execution cost.

Furthermore, since most of the times TCs are nightly executed, they often claimed the impossibility of being able to execute the entire TSs. In those cases, priority to fast fault detection needs to be given to start the debugging process as soon as possible.

To address the above-mentioned challenges, we proposed a search-based approach to cost-effectively generate and prioritize reactive TCs for testing industrial CPSs. First, together with our industrial partners, we defined a fitness function with four cost-effectiveness measures (i.e., objectives) including RC, TC similarity (TCS), prioritization-aware TCS, and TET. Second, we designed one crossover operator and two mutation operators (including a mutation operator at a TS level and a mutation operator at a TC level). The defined fitness functions and designed crossover and mutation operators were incorporated with five multi-objective search algorithms [e.g., nondominated sorting genetic Algorithm II (NSGA-II)] and evaluated using four industrial case studies for different domains (e.g., UAVs). Note that random search (RS) is also used as a baseline algorithm for the evaluation. The evaluation results showed that NSGA-II improved the performance of the rest of the selected algorithms on average between 11.93% and 47.07%. In addition, we recommend practitioners to apply our approach with the crossover rate of 0.2 and the mutation rate of $2/N$, which can achieve the best performance.

This paper is an extension of a conference paper [13] and the key improvements include

1) The approach has been extended to support test prioritization in addition to TC generation by adapting the defined mutation and crossover operators and defining a new objective (i.e., prioritization-aware TCS);
2) The objective of TET was improved to include the TET based on the prioritization of the TCs;
3) A new related work section has been added to position our research with the state-of-the-art of CPS testing;
4) A new research question (RQ) is added to empirically evaluate the performance of our approach with crossover and mutation operators with five crossover rates (CXR) and three mutation rates (MR) (in total 15 combinations);
5) The empirical evaluation has been largely extended by employing new evaluation metrics (e.g., RC) and four new multi-objective search algorithms (e.g., improved strength pareto evolutionary algorithm); and
6) A set of lessons learned (LLs) is shared based on the collaboration with our industrial partners, which can be used as a guidelines for future practitioners.

The rest of the paper is structured as follows. Section II presents an overview of the background. Section III positions this paper with relevant work. The test generation and prioritization approach is presented in Section IV followed by the empirical evaluation (see Section V). The LLs are discussed in Section VI, and Section VII concludes the paper.

II. Background

A. Cyber-Physical Systems Testing

As developing a CPS prototype is often costly, the use of simulation-based testing for CPSs is increasing. Simulation is one of the most frequently used techniques for testing system models in domains where software interacts with physical

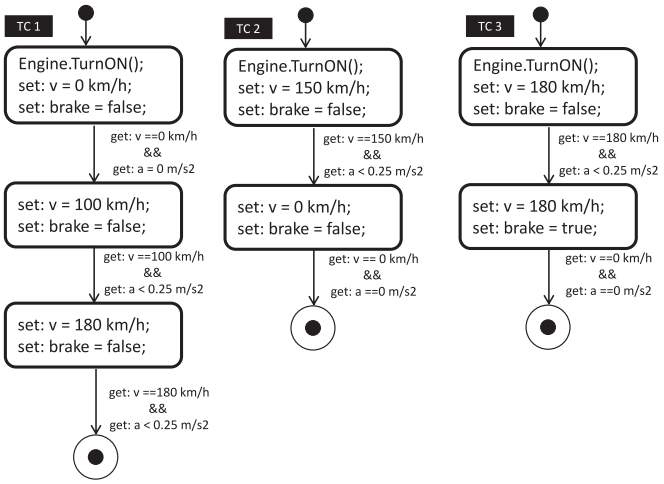


Fig. 1. Example of three reactive test cases for the cruise control system testing of a vehicle.

processes such as CPSs [8]. CPS models are heterogeneous due to encompassing software, networks, and parallel physical processes. These models, therefore, provide an accurate representation of the real world and continuous dynamics [4], [8]. Different simulation tools have been employed to test CPSs using simulation including MATLAB/Simulink [8], [9], [14], and a combination of system C and open dynamics engine [15]. Moreover, as CPSs are involved in several engineering domains, it is very common to use different simulation environment integrated by a cosimulation engine. Although this integrated CPS model presents an important advance in terms of engineer flexibility, it increases the test and simulation time due to requiring the transfer of data between tools. In addition, CPSs simulation often involves the use of complex mathematical models to represent the continuous dynamics of the physical layer. Consequently, computer resources are allocated by the solvers used by the simulation tools, incrementing the simulation time. In addition, CPSs are often tested at different levels such as MiL, SiL, and HiL [12], [16].

B. Reactive Test Cases

Test reactivity is known as the capacity of the test system to react on the outputs of the system under test, verdicts data, or internal signals of the test oracle [17]. In the context of CPSs, a reactive TC is referred to a state-based TC that monitors the state of the system at runtime, and it reacts on them by changing the states when a transition is triggered based on the obtained runtime system state. For instance, as shown in Fig. 1, the test case TC1 will change from state 1 to state 2 when a transition obtained from the system (i.e., get $v = 0$ km/h and get $a = 0$ m/s^2) occurs. As compared with reactive TCs, traditional TCs specify all the inputs and produce the outputs statically without monitoring and interacting with systems at runtime.

As an example, consider three reactive TCs depicted in Fig. 1, which aim at testing the cruise control system of a car. With this objective, the TCs can turn ON the engine of the car, set the speed of the car (v), or push the brake pedal to reduce the speed of a car should the driver wish. The TC will change the state when a transition is triggered [i.e., in this case, the speed has been stabilized with a minimum acceleration (a) of the car]. In the illustrated example, the first reactive TC (i.e., TC1) contains three states. Unlike TC1, reactive TCs 2 and 3 (i.e., TC2 and TC3) have two states. When considering these three TCs, it can be easily deduced that TC1 tests acceleration functionalities, TC2 tests deceleration functionalities, and TC3 tests functionalities related to the braking system.

III. Related Work

The interest in TC generation for CPSs has increased in the last few years. Several approaches proposed different TC generation approaches for CPSs based on MBT combined with conformance testing [18]–[22]. However, MBT techniques suffer from scalability and are usually expensive, time-consuming, or infeasible to apply [7]. To overcome these problems, other works rely on search-based techniques guided by a fitness function to generate TCs for CPSs [8], [23], [24]. Their fitness functions are obtained by using simulation models of the CPSs to search for "worst-case scenarios." While these approaches have been found to be effective, a drawback of them is that the execution time of the algorithms is high because simulating the system in each iteration might be too costly. Other approaches consider improving the verification and validation stages of CPSs by using runtime verification techniques [3] or by automatically generating test system instances for configurable CPSs [9].

Previously, some works focused on the generation of reactive TCs. Zander captured the reactivity of the system with test oracles and later employed a model-based approach to generate one TC to test each requirement [25]. Lehmann proposed a tool named time partition testing for the elaboration of model-based reactive TCs and it automatically generated executable TCs for different simulation tools (e.g., Simulink) [26]. These works focused on testing requirements, but they did not consider other objectives, such as the TET or test similarity. In addition, in all of them, the process of generating TCs is semi-automatic as they all need to specify some reactivity behaviors.

As for test optimization of CPSs, in our previous work, we proposed a TC selection method [12] as well as a test prioritization approach employing reactive TCs for configurable CPSs based on weighted search algorithms. The prioritization approach involved two objectives [i.e., TET and fault detection capability (FDC)] [10]. The FDC objective used in [10] requires historical data, which is obtained after the TCs have been executed for some time. Although this is an advantage when several configurations have been tested, it might need a long start-up to be effective.

IV. Search-Based Test Case Generation and Prioritization Approach

A. Cost-Effectiveness Measures

Four cost-effectiveness measures have been defined to guide the search toward generating optimal reactive TCs, i.e., RC,

TCS, prioritization-aware TCS (effectiveness), and TET (cost). Each measure is presented in detail as follows.

1) Requirements Coverage: Reactive TCs are typically employed for functional testing at the system level, meaning that functional requirements must be taken into account when generating TCs. For this reason, the RC measure is defined as the first effectiveness measure, which considers the number of requirements covered by a specific TS with respect to the total number of requirements of a CPS. The RC is calculated following (1), where $|\text{FR} \lessdot sk|$ refers to the number of functional requirements covered by a solution, while $|\text{FR}|$ denotes the total number of functional requirements in the CPS. Notice that RC is developed for each system with a script function by linking each requirement with a specific property of state or sequence of states. For instance, if the acceleration functionality of the cruise control was tested, a state would set the car to 0 km/h, whereas the proceeding state would set the speed to more than 150 km/h. A coverage matrix is built by linking each TC with each of the covered requirement. The RC is later trivially determined by processing this matrix. We obtained the requirements from our industrial partners, which are captured with natural language, e.g., "the system warns the driver with an acoustic alarm if the actual distance to the vehicle ahead is less than (current speed/3.6) * time":

$$\text{RC}_{sk} = \frac{|\text{FR} \lessdot sk|}{|\text{FR}|}. \tag{1}$$

2) Test Case Similarity: Existing literature has shown that a set of diverse TCs has a higher chance to detect faults [11]. For this reason, the TCS was defined as the second effectiveness measure based on the Hamming distance, which is a widely used similarity measure [27]. More specifically, TCS measures the distance between two TCs and returns a value within the range [0, 1]. A TCS value with 0 denotes that both TCs are identical, whereas when the TCS returns a value of 1, both of the TCs are considered as totally different. In addition to that, it can be followed in the context of reactive TCs that a reactive TC with more states is more likely to find errors than a TC with less states. With this concern in mind, the number of states present is taken into account when calculating TCS.

The TCS of two TCs (i.e., TC_a and TC_b) is calculated, as expressed in (2), where $|S_a|$ and $|S_b|$ are the number of states of each of the TCs and $|ss|$ is the number of stimuli signals for the CPS. $ss_{j_{\text{TC}_{a\,i}}}$ is the jth signal's value for the a TC's ith state and $ss_{j_{\text{TC}_{b\,i}}}$ is the jth signal's value for the b TC's ith state. In addition, $\max_{ss_j}$ is the maximum value and $\min_{ss_j}$ is the minimum value that the jth signal can have. It is considered that these maximum and minimum values will always be different, and thus, the interval between $\max_{ss_j}$ and $\min_{ss_j}$ will not be 0. Finally, MaxStates refers to the maximum number of states a TC can have, which is predefined by the test engineer:

$$\text{TCS}(\text{TC}_a, \text{TC}_b) = \frac{\sum_{i=1}^{\min(|S_a|,|S_b|)} \sum_{j=1}^{|ss|} \frac{\text{abs}(ss_{j_{\text{TC}_{a\,i}}} - ss_{j_{\text{TC}_{b\,i}}})}{\text{abs}(\max_{ss_j} - \min_{ss_j})} / |ss|}{\text{MaxStates}}. \tag{2}$$

Consider as an example, the TCS between TC1 and TC2 from sample Fig. 1. Let us assume that the system has two stimuli signals (i.e., v and brake), the maximum speed that the system can be set to is 180 km/h, and the maximum number of states is 3. Thus, the TCS between TC1 and TC2 would be calculated as follows:

$$\text{TCS}(\text{TC1}, \text{TC2}) = ((\text{abs}(0-150)/180 + \text{abs}(0-0)/1)/2 + (\text{abs}(100-0)/180 + \text{abs}(0-0)/1)/2)/3 = 0.2315.$$

Furthermore, given a solution sk with NTC number of TCs, the average similarity function can be measured by calculating the average TCS values for each TC pair. This is obtained in the following equation, where NTC is the number of TCs in sk, i is the ith TC, j is the jth TC, and $\text{NTC} \times (\text{NTC}-1)/2$ is the total number of TC combinations in sk:

$$\text{Sim}_{sk} = 1 - \frac{\sum_{i=1}^{\text{NTC}} \sum_{j=i+1}^{\text{NTC}} \text{TCS}(\text{TC}_i, \text{TC}_j)}{\text{NTC} \times (\text{NTC}-1)/2}. \tag{3}$$

3) Prioritization-Aware Test Case Similarity: In other works, historical data are employed to prioritize TCs (e.g., [10]). However, in this study, TCs are being prioritized while they are being generated, which means that historical data are not available. For this reason, since diversifying the execution of TCs might imply having a higher chance of detecting faults, a new objective named prioritization-aware similarity (PSim) was included. This objective measures the average similarity of each of the TCs with its preceding TCs (i.e., TCs that were prioritized before). This permits guiding the search toward a prioritized TS where the TCs in the initial positions of the suite are more different with one another. Given a prioritized test suite $\text{PTS} = \{\text{TC}_1, \text{TC}_2, ..., \text{TC}_{\text{NTC}}\}$, the following equation shows how the PSim objective is measured:

$$\text{PSim}_{sk} = 1 - \frac{\sum_{i=2}^{\text{NTC}} \frac{\sum_{j=1}^{i-1} \text{TCS}(\text{TC}_i, \text{TC}_j)}{i-1}}{\text{NTC}-1}. \tag{4}$$

4) Test Execution Time: The time for executing a TS is essential in the CPS testing context [10], [12]. This is, to a large extent, caused by the high computational resources that simulation solvers consume to compute complex mathematical models related to the physical layer. Thus, the TET is defined as a cost measure for the generation of TCs for the CPS context. Given a solution sk of NTC TCs, each TC i has NS_{TC_i} number of states, the TET of sk is calculated as

$$\text{TET}_{sk} = \sum_{i=1}^{\text{NTC}} (\text{time}(S_{\text{NS}_{\text{TC}_{i-1}}}, S_{1_i}) + \sum_{j=2}^{\text{NS}_{\text{TC}_i}} \text{time}(S_{j_i}, S_{j-1_i})). \tag{5}$$

An idiosyncrasy of reactive TCs is that their TET varies depending on the previously prioritized TC. Thus, it is important to highlight that the first part in (5) refers to the initialization time of the TC, which was not considered in the conference version paper. This is because in the conference version paper, test prioritization was not taken into account. Since in this case, apart from test generation, test prioritization is also performed, the time required to initialize the TC is taken into account [i.e.,

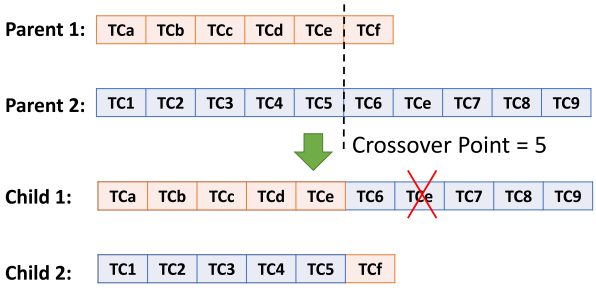


Fig. 2. Crossover operator example for two test suites of 6 and 10 test cases, having the crossover point at the fifth place.

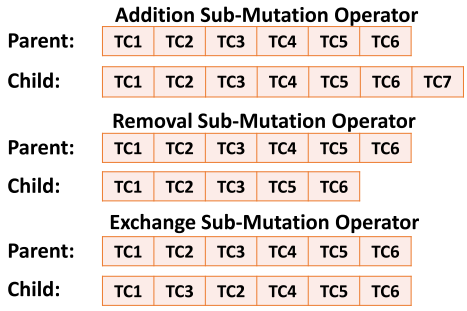


Fig. 3. Submutation operators at a test suite level.

time between the last state of the previously executed TCs (i.e., $S_{\mathrm{NS}_{\mathrm{TC}_{i-1}}}$) and the first state of the TC (i.e., S_{1_i})].

Notice that the function time, which sets the time required by the system to change from one state to another, is system specific. This means that before launching the test generation approach, the test engineer must specify how the function time is computed. For instance, in the case of the cruise control example, where the TCs are depicted in Fig. 1, the time function between two states would be $\mathrm{time}(s1, s2) = a_c \times \Delta V$, where ΔV is the difference between both state speeds and a_c is the acceleration coefficient, which changes if the car is accelerating, decelerating, or braking.

B. Solution Representation

A solution in this context is a TS composed by at least one TC, i.e., $\mathrm{TS} = \{\mathrm{TC1}, \mathrm{TC2}, ..., \mathrm{TC}_N\}$, where N is the total number of TCs in TS. Accordingly, a TC in our context is a reactive TC. Each of these TCs is composed by a set of states (S), (i.e., $\mathrm{TC}_i = \{S_1, S_2, ..., S_{N_{\mathrm{TC}_i}}\}$, where N_{TC_i} is the number of states that the ith TC is composed of). Each state S has a predefined set of stimuli signals (ss) that must be connected to the simulation model of the CPS. These stimuli signals are based on the simulation model and can be of different types (i.e., Boolean, integer, or double), and each of them has a maximum and a minimum value. Typically, CPSs TSs are composed of around 100 reactive TCs [10].

C. Crossover Operator

The implemented crossover operator exchanges the TCs between two different solutions. Given two TSs, a randomly generated crossover point is selected, in the range [1, N], where N is the number of TCs related to the smallest TS. When the crossover point is selected, two children are generated combining TCs between both TSs. A single point crossover was used but the algorithm can also be configured to use multipoint crossover.

Consider as an example two parent TSs, as shown in Fig. 2. One of them contains six TCs, whereas the other ten. Thus, the crossover function will randomly select a crossover point in the range [1, 6]. In the illustrated example, the crossover point is number 5. Child 1 maintains the first 5 TCs of parent 1, while it inherits the sixth to tenth TCs of parent 2. On the other hand, child 2 maintains the first 5 TCs of parent 2, while it inherits the sixth TC of parent 1. However, notice that one of the TCs corresponding to the TS of parent 1 and parent 2 is the same (i.e., TCe). Child 1 inheres this TC from parent 2, and thus, the TC is repeated in the TS. When this happens, the TC in a later position is removed.

D. Mutation Operators

The mutation operators are designed from two levels. The first level is the TS level, whereas the second one is the TC level. The mutation operator at the TS level randomly mutates TCs by adding, removing, or exchanging them from the TS. The mutation operator at the TC level mutates the states by modifying them.

1) Mutation Operator at the Test Suite Level: For the mutation operator at the TS level, three submutation operators have been developed. When the mutation operator at the TS level is selected, one of these submutation operators is randomly chosen by the algorithm. The first submutation operator consists of the addition of a new TC into the TS. When a new TC is added, this operator randomly decides its position in the TS as well as the number of states that this TC must have. When the number of states is decided, it randomly selects the values of their stimuli signals based on the type and the maximum and minimum values they can be set to. The second submutation operator consists of the removal of a TC from the TS. It randomly selects the TC to be removed from the TS and a child TS is generated without the selected TC. A third submutation operator has been developed at the TS level to account for the TC prioritization, which consists of exchanging the position of two TCs. When this submutation operator is selected, it randomly selects two TCs and exchanges their position. Fig. 3 illustrates the submutation operators for the TS level. As for the first submutation operator, a new TC is added in the last position of the TS. As for the second submutation operator, the fourth TC is removed from the TS. Finally, the exchange submutation operator selects the second and third TCs and their positions are swapped.

2) Mutation Operator at the Test Case Level: The mutation operator at the TC level operates within the states that certain TCs have. At this level, four submutation operators have been developed, consisting of state addition, state removal, exchange of states, and change of variable. When the mutation operator at the TC level is selected, one of these submutation operators is randomly chosen.

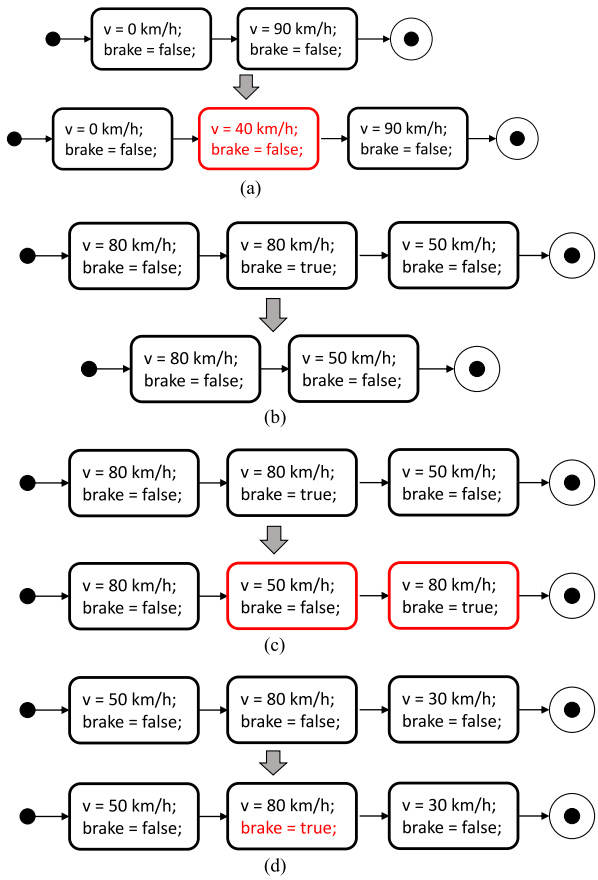


Fig. 4. Submutation operators for reactive test cases at a test case level. (a) Addition sub-mutation operator for the test case level. (b) Removal sub-mutation operator for the test case level. (c) Exchange sub-mutation operator for the test case level. (d) Change of variable sub-mutation operator for the test case level.

For the state addition operator, a TC from the TS is randomly chosen and a new state is added in a random position of that TC, as exemplified in Fig. 4(a). For the state removal operator, a TC is randomly chosen from the TS and one of its states is randomly removed, as illustrated in Fig. 4. For the state exchange operator, a TC is randomly chosen from the TS. Later, two states are randomly selected and their positions are exchanged, as depicted in Fig. 4. Finally, the change of variable operator randomly chooses one of the stimuli signals of a TC and its value is changed according to its type and maximum as well as minimum values, as shown in Fig. 4.

Note that the process of generating such reactive TCs is automatic with our approach. More specifically, our approach first randomly generates an initial set of TSs (i.e., population) based on the inputs of the system (i.e., stimuli signals). Each TS is composed of a set of randomly generated reactive TCs, e.g., the three TCs depicted in Fig. 1. Afterward, our approach applies the defined crossover and mutation operators on the population with the aim to produce new offspring TSs. For instance, the speed in the state 2 of the TC 1 (see Fig. 1) can be mutated to 50 km/h, and thus, a new TC can be generated. The cost-effectiveness measures defined in Section IV-A are defined to assess the quality of the generated TSs.

E. Tool Support

In our case, we first employed MATLAB to implement our defined fitness function and incorporate with multi-objective search algorithms. Second, we employed the tool FeatureIDE [28] to specify the format of generated TCs (i.e., number and types of stimuli signals) followed by using Simulink to execute the generated TCs, due to its wide use in the industry [7], [8]. We further implemented a tool to transform abstract TCs into executable Simulink TCs. Note that all the above-mentioned tools have been integrated as a tool chain, which is in the ongoing process to be applied in the practice of our several industrial partners.

V. Empirical Evaluation

A. Research Questions

As discussed in Section III, we first performed a literature review to position our work with the state-of-the-art. The outcome of the literature review indicated that the generation and prioritization of reactive TCs for CPS testing have not been widely explored in the state-of-the-art. Thus, the key focus of this paper is to propose a cost-effective approach for generating and prioritizing reactive TCs. To evaluate our approach, we aimed to answer three RQs, which are detailed as follows.

1) *RQ1:* Are the selected multi-objective algorithms cost-effective when compared to RS for solving the TC generation and prioritization problem?

RQ1 is defined as a sanity check to verify that our test generation and prioritization problem is nontrivial to solve. To address this RQ, we compared our approach with different multi-objective search algorithms with RS, which is the baseline for a randomized algorithm [29]. If our approach can significantly outperform RS, we will further assess which multi-objective search algorithms can assist our approach in achieving the best performance (i.e., RQ2).

2) *RQ2:* Which of the selected multi-objective algorithms fares best when solving the TC generation and prioritization problem?

RQ2 is defined to identify the best multi-objective search algorithm to generate and prioritize reactive TCs for testing CPSs, which will be integrated into our approach and used in practical settings. To tackle this RQ, we chose five state-of-the-art multi-objective search algorithms (e.g., NSGA-II), which have been widely applied in the existing work [30]. Once the best multi-objective search algorithm is obtained, we will study and investigate if different CXR and MR can affect the performance of our approach, which is the key motivation of RQ3.

3) *RQ3:* How do the designed crossover and mutation operators with different CXR and MR affect the performance of our approach?

RQ3 is defined to assess the performance of the crossover and mutation operators along with different rates. To deal with this RQ, we chose in total five CXR (i.e., 0, 0.2, 0.5, 0.8, and 1) and three MR (i.e., $1/N$, $2/N$, and $5/N$) based on the guidelines from [31]. Thus, in total 15 combinations were obtained for each of the selected four case studies, e.g., crossover rate with 0.5 and mutation rate with $1/N$.

TABLE I
KEY CHARACTERISTICS OF THE CASE STUDIES

Case study	Reqs	Stimuli signals	Blocks	Depth
DC engine	25	4 (2-B, 2-I)	257	3
UAV	22	10 (6-B, 4-I)	843	4
ACC	10	7(4-B, 3-I)	415	5
Tank	9	3 (3-I)	112	4

B. Experimental Setup

1) Case Studies: Four case studies that were modeled in MATLAB/Simulink were employed in the proposed empirical evaluation. An open source system involving the automatic control of a dc engine with safety-critical functionalities was selected as a first case study. The second case study was the model of a UAV. The third case study was the adaptation of an industrial system from Daimler AG concerning the adaptive cruise control (ACC) of a car. The last case study consisted of an industrial tank system. All these case studies were defined together with our industrial partners. Table I reports the key characteristics of the selected case studies. Specifically, the *Reqs* column specifies the number of requirements of the systems. The *Stimuli Signals* column is the number of stimuli signals of their Simulink models (B means the number Boolean signals and I means the number of integer signals). The last two columns are related to the Simulink models of the systems; the *Blocks* column is related to the number of blocks that the systems has, whereas the column *Depth* refers to the number of hierarchical levels of the system model.

2) Parameters Configurations: Five search algorithms were selected: NSGA-II, strength pareto evolutionary algorithm 2 (SPEA2), multi-objective evolutionary algorithm based on decomposition (MOEA/D), pareto envelope-based selection algorithm II (PESA-II), and nondominated sorting genetic algorithm III (NSGA-III). The designed crossover and mutation operators were integrated into the selected algorithms.

For RQ1 and RQ2, as recommended by one of the most commonly applied multi-objective optimization Java framework jMetal [32], we set the crossover rate as 0.9, the population size was 100, and the number of fitness evaluations was 100 000. The mutation probability is $1/N$, N being the number of TCs for the mutation operator at the TS level, N being the number of states for the mutation operator at the TC level and three first mutation suboperators, and N being the number of stimuli signals for the mutation operator at the TC level and the fourth mutation suboperator. In addition, for RQ1, each algorithm was run 100 times to account for random variations as recommended by Arcuri and Briand [29].

As for RQ3, as mentioned in Section V-A, we chose in total 15 combinations, which were obtained based on the guidelines from [31]. We chose in total five CXR (i.e., 0, 0.2, 0.5, 0.8, and 1) and three MR (i.e., $1/N$, $2/N$, and $5/N$), leading to in total 15 combinations, e.g., crossover rate with 0.5 and mutation rate with $1/N$. We further evaluated our approach along with the best multi-objective search algorithm (based on the results from RQ2) using the four case studies (see Table I). More specifically, we first compared the performance of each pair of MR under the same crossover rate, e.g., crossover rate with 0 and mutation rate with $1/N$ vs. crossover rate with 0 and mutation rate with $2/N$. For each crossover rate, we chose the MR with the best performance followed by comparing different CXR with the best MR.

3) Evaluation Metrics: Based on the guide [30], the hypervolume (HV) quality indicator was selected as the evaluation metric for the empirical evaluation. To be specific, HV measures the volume in the objective space covered by the produced solutions [32] with the range from 0 to 1 and a higher value of HV denotes a better performance of the algorithm. It is important to note that HV has been applied to assess similar multi-objective test optimization approaches (e.g., TC generation approach [23]). In addition to the HV quality indicator, the four fitness values of each selected objectives were measured, with the aim of measuring the ability of each algorithm in terms optimizing each objective.

4) Statistical Analysis: As recommended by Arcuri and Briand, the Mann–Whitney U test (i.e., significance test) was used to determine the significance of the results produced by different algorithms, and the significance level was set to 95%, whereby there is a statistically significant difference between the results of two algorithms if the p-value is less than 0.05. To determine the difference existing between two algorithms, the Vargha and Delaney statistics was used to calculate the $\hat{A}_{12}$ measure [29], [33].

C. Results and Analysis

This section discusses the key results and observations.[1] Notice that in our conference version paper, we only compared NSGA-II with RS as the rest of algorithms (i.e., SPEA2, MOEA/D, PESA-II, and NSGA-III) were not used. As shown in Fig. 5 and corroborated by means of statistical tests, according to the HV quality indicator, all the algorithms significantly outperformed RS, although there were some exceptions. For the ACC case study, RS outperformed with statistical significance (according to the Mann–Whitney U test) SPEA2, PESA-II, and NSGA-III, and in the UAV case study, RS also outperformed NSGA-III. For the ACC case study, regarding the remaining objectives, all the algorithms significantly outperformed RS with the exception of the similarity and the PSim, where RS significantly outperformed SPEA2 and PESA-II, according to the Mann–Whitney U test. In addition, RS significantly outperformed MOEA/D in three out of four case studies for the TET time objective. Nevertheless, NSGA-II significantly outperformed RS since all the p-values are less than 0.05 for each case study and each objective. As for the improvements, Table II shows the average percentage improved for the HV, as well as each of the objectives, by NSGA-II with respect to RS, which was taken as the baseline algorithm. On average, the HV quality indicator was improved by 49.25%, the RC by 51.74%, the TET by 62.96%, the TCS by 29.86%, and the PSim by 30.65%. NSGA-II was able to outperform RS for on average 43.80% in terms of each objective.

[1]Due to the page limitation, we only provide summarized key results in the paper and detailed results can be found in https://sites.google.com/view/tii2017

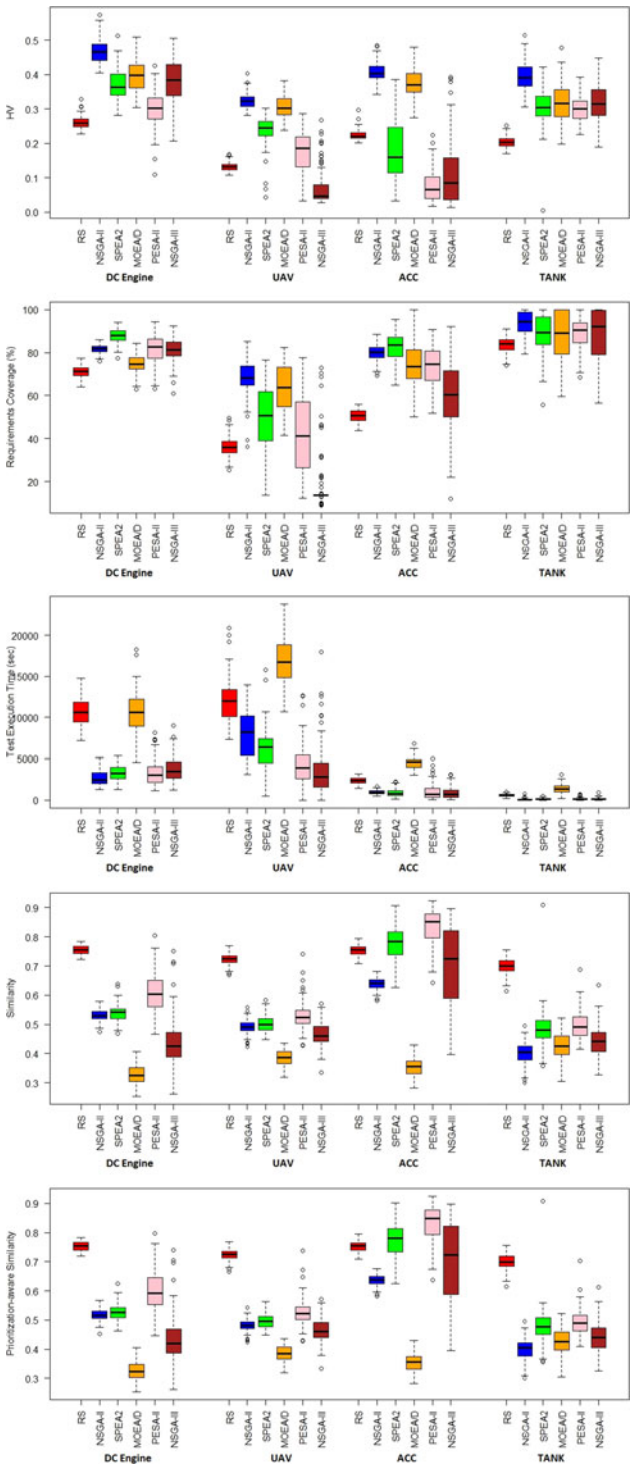


Fig. 5. Distribution of the selected evaluation metrics for all the runs in each case study.

The second RQ assessed which of the selected algorithms performed the best. Fig. 5 shows the distribution of the HV indicator, as well as each objective (e.g., TET), for the 100 algorithm runs performed in each case study. Notice that for the HV indicator, the higher the value, the better the algorithm performance. According to the HV quality indicator, NSGA-II outperformed the rest of the algorithms. Table III reports

TABLE II
AVERAGE PERCENTAGE OF IMPROVEMENT OF NSGA-II COMPARED WITH RS FOR EACH OBJECTIVE AND EACH CASE STUDY

	HV	RC	TET	Similarity	PrioSim
DC engine	44.37%	36.76%	75.02%	29.54%	31.13%
UAV	59.28%	50.04%	32.92%	32.23%	33.45%
ACC	44.95%	59.47%	60.17%	15.05%	15.32%
Tank	48.42%	60.71%	83.74%	42.60%	42.69%
Average	49.25%	51.74%	62.96%	29.86%	30.65%

TABLE III
HV QUALITY INDICATOR IMPROVEMENT PERCENTAGE OF NSGA-II WITH RESPECT TO THE REMAINING SEARCH ALGORITHMS

	DC engine	UAV	ACC	TANK	Avg.
RS	44.37%	59.28%	44.95%	48.42%	49.25%
SPEA2	20.98%	26.90%	55.58%	23.01%	31.62%
MOEA/D	15.57%	4.66%	8.19%	19.29%	11.93%
PESA-II	35.80%	47.68%	80.60%	23.89%	46.99%
NSGA-III	18.48%	77.73%	72.42%	19.67%	47.07%

the average percentage of improvement of NSGA-II with respect to the remaining selected algorithms for the 100 algorithm runs. On average, for the HV quality indicator, NSGA-II outperformed SPEA2 in 31.62%, MOEA/D in 11.93%, PESA-II in 46.99%, and NSGA-III in 47.07%. This was further corroborated by means of statistical tests employing the Mann–Whitney U test.[2] Furthermore, NSGA-II significantly outperformed the other algorithms based on the results of Mann–Whitney U test (all the p-values are less than 0.05). Conversely, for the HV indicator, the algorithm showing the worst performance was NSGA-III. Apart from the HV quality indicator, for the remaining objectives, in general, NSGA-II was also the best algorithm. However, as shown in Fig. 5 as well as in the statistical tests, there were some exceptions. For instance, for the dc engine and the ACC case studies, the SPEA2 algorithm outperformed with statistical significance the NSGA-II algorithm for the RC objective. In addition, for the TET, PESA-II and the NSGA-III also outperformed the NSGA-II algorithm in the UAV case study. Furthermore, except for the tank case study, for the similarity, and the PSim, the MOEA/D performed better than the NSGA-II. However, the MOEA/D algorithm was outperformed by the rest of algorithms for the TET objective.

RQ3 aimed at investigating the performance of our approach with different CXR and MR. Table IV summarizes the statistical test results of comparing each pair of MRs (e.g., 1/N vs. 2/N) for each CXR. Note that $\hat{A}_{12}$ lower than 0.5 means that the MR in the left performs better than the MR in the right and vice versa. A boldface value in Table IV indicates there is a significant difference based on the Mann–Whitney U test (i.e., p-value $\leq$ 0.05). Based on the results from Table IV, the MR with 2/N significantly outperformed 5/N for all the case studies with respect to all the five CXR except for one case (i.e., 2/N outperformed 5/N but not significantly for the CXR with 0.5 of the tank case

[2]https://sites.google.com/view/tii2017/table

TABLE IV
STATISTICAL TESTS RESULTS SUMMARY FOR HV QUALITY INDICATOR AND DIFFERENT MUTATION RATES

		DC eng.	UAV	ACC	TANK
CXR	MR	$\hat{A}_{12}$	$\hat{A}_{12}$	$\hat{A}_{12}$	$\hat{A}_{12}$
0	1/*N* vs. 2/*N*	0.582	**0.657**	0.622	**0.801**
	1/*N* vs. 5/*N*	**0.266**	0.478	**0.154**	0.421
	2/*N* vs. 5/*N*	**0.122**	**0.244**	**0.113**	**0.082**
0.2	1/*N* vs. 2/*N*	**0.666**	**0.647**	0.544	**0.867**
	1/*N* vs. 5/*N*	0.377	0.421	**0.174**	**0.653**
	2/*N* vs. 5/*N*	**0.242**	**0.247**	**0.167**	**0.242**
0.5	1/*N* vs. 2/*N*	0.472	0.579	0.528	**0.843**
	1/*N* vs. 5/*N*	**0.175**	0.402	**0.273**	**0.750**
	2/*N* vs. 5/*N*	**0.192**	**0.328**	**0.247**	0.389
0.8	1/*N* vs. 2/*N*	**0.658**	0.588	0.606	**0.820**
	1/*N* vs. 5/*N*	0.378	0.444	**0.231**	0.634
	2/*N* vs. 5/*N*	**0.211**	**0.344**	**0.141**	**0.236**
1	1/*N* vs. 2/*N*	0.613	0.621	**0.697**	**0.903**
	1/*N* vs. 5/*N*	**0.340**	0.453	**0.248**	**0.770**
	2/*N* vs. 5/*N*	**0.262**	**0.334**	**0.119**	**0.264**

TABLE V
STATISTICAL TESTS RESULTS SUMMARY FOR HV QUALITY INDICATOR AND DIFFERENT CROSSOVER RATES WHEN MUTATION PROBABILITY IS *N*/2

	DC eng.	UAV	ACC	TANK
CXR	$\hat{A}_{12}$	$\hat{A}_{12}$	$\hat{A}_{12}$	$\hat{A}_{12}$
0 vs. 0.2	**0.73**	**0.75**	**0.67**	0.64
0 vs. 0.5	**0.71**	**0.71**	**0.66**	0.53
0 vs. 0.8	**0.81**	**0.73**	**0.76**	0.65
0 vs. 1	**0.80**	**0.76**	**0.79**	0.57
0.2 vs. 0.5	0.48	0.48	0.51	0.41
0.2 vs. 0.8	0.62	0.50	0.61	0.51
0.2 vs. 1	0.60	0.53	**0.65**	0.43
0.5 vs. 0.8	0.63	0.53	0.58	0.60
0.5 vs. 1	0.62	0.55	0.62	0.52
0.8 vs. 1	0.48	0.53	0.56	0.42

study). Moreover, the MR 2/*N* also outperformed 1/*N* for all the cases, and such better performance was statistically significant in 10 out of 20 cases. Therefore, we can conclude that the MR with 2/*N* managed to achieve the best performance as compared with 1/*N* and 5/*N*.

Furthermore, we evaluated each pair of CXRs with the best MR of 2/*N*. Table V reports the statistical results. The results showed that the CXRs with 0.2, 0.5, 0.8, and 1 performed better than CXR with 0 for all the case studies, in three of them with statistical significance. Such results indicated that the designed crossover operator can help our approach to find better optimal solutions. Moreover, we also observed there was no statistical significance in general for the performance of CXRs with 0.2, 0.5, 0.8, and 1, meaning that any of them can be applied along with our approach. Since running time is an important perspective when evaluating a search-based approach [23], we also reported the average running time (with 30 runs for each case study) for the CXRs of 0.2, 0.5, 0.8, and 1 together with the MR of 2/*N* as 632.2, 807.3, 1012.8, and 1163.5 s, respectively. To determine if there exists significant difference among these running time, we performed the Mann–Whitney *U* test and the results showed that with the MR of 2/*N*, the CXR of 0.2 took significantly less running time than the remaining CXRs. Thus, we can conclude that the CXR of 0.2 and MR of 2/*N* can assist our approach in achieving the best performance, which should be recommended to the future practitioners.

The average and standard deviation values related to the HV quality indicator for the 15 combinations are shown in Table VI. As it can be observed, the mutation rate of 2/*N* showed higher HV average values in 19 out of 20 cases. The average value of the HV quality indicator for the mutation rate of 1/*N* was 0.39, for the mutation rate of 2/*N* was 0.41, and for the mutation rate of 5/*N* was 0.38.

D. Overall Discussion

The first RQ aimed to answer whether the test generation and prioritization problem for the CPS context was not trivial to solve. To this end, the selected algorithms were compared with RS. Results indicated that, in general, the selected multi-objective search algorithms outperformed RS. Specifically, NSGA-II outperformed RS for all the four objectives and the HV in the four case studies with statistical significance. Thus, it can be concluded that the problem to solve is not trivial.

The second RQ aimed to identify which of the algorithms performed the best when solving the test generation and prioritization problem. Overall, considering the HV quality indicator, which measures the volume in the objective space, NSGA-II showed the best performance. However, MOEA/D showed the best performance for both defined similarity measures in three out of four case studies. This means that MOEA/D is good at generating dissimilar TCs. However, for the rest of objectives (i.e., TET and RC) as well as for the HV, NSGA-II was the best algorithm. The better performance of the NSGA-II could be that as compared to SPEA2, MOEA/D, and PESA-II, NSGA-II sorts the produced solutions into several nondominated Pareto fronts and chooses the solutions to be included for the next generation from the best fronts until the population size is met. Such mechanism can help to choose the solutions only with the best quality for producing the offspring solutions. Thus, the convergence to the optimal solutions achieved by NSGA-II is relatively fast. Moreover, NSGA-II defined a metric named crowd distance to measure the diversity of solutions, which is not the case for the other algorithms (i.e., SPEA2, MOEA/D, PESA-II, and NSGA-III). Note that the diversity of solutions (i.e., generated and prioritized TCs) is very important since we aim at testing CPSs with diverse inputs (i.e., states and stimuli signals) and we have defined two objectives dedicated to measure the similarity of TCs (i.e., similarity and PSim, Section IV-A). Thus, NSGA-II with a measure of solution diversity can manage to assist our approach in achieving the best performance as compared with the other four multi-objective search algorithms.

RQ3 aimed to study which CXR and MR could help our approach to achieve the best performance. The results showed that the mutation rate 2/*N* outperformed 1/*N* and 5/*N*. Such interesting observation can be explained that in our case, optimal solutions are distributed in the search space, which requires mutation operators exploring the search space toward finding optimal solutions. Thus, a higher mutation rate can help to obtain optimal solutions quickly. However, the mutation rate 5/*N* seemed to be too high and might lead to distort optimal solutions rather than

TABLE VI
AVERAGE AND STANDARD DEVIATION HV VALUES FOR THE 30 RUNS OF THE NSGA-II WITH DIFFERENT CROSSOVER AND MUTATION OPERATOR RATES

		DC eng.		UAV		ACC		TANK		Average	
CXR	MR	Avg.	σ	Avg.	σ	Avg.	σ	Avg.	σ	Avg.	σ
0	1/*N*	0.433	0.035	0.307	0.026	0.387	0.024	0.379	0.038	0.377	0.031
	2/*N*	0.443	0.028	0.320	0.023	0.396	0.024	0.422	0.033	0.395	0.027
	5/*N*	0.405	0.019	0.302	0.014	0.355	0.023	0.368	0.023	0.358	0.020
0.2	1/*N*	0.450	0.026	0.330	0.025	0.410	0.028	0.379	0.042	0.392	0.030
	2/*N*	0.470	0.037	0.342	0.021	0.412	0.026	0.437	0.031	0.415	0.029
	5/*N*	0.438	0.024	0.323	0.021	0.377	0.024	0.400	0.037	0.385	0.027
0.5	1/*N*	0.469	0.032	0.333	0.020	0.410	0.028	0.382	0.029	0.398	0.027
	2/*N*	0.466	0.032	0.340	0.027	0.416	0.034	0.428	0.041	0.413	0.034
	5/*N*	0.432	0.025	0.325	0.021	0.387	0.023	0.411	0.035	0.389	0.026
0.8	1/*N*	0.461	0.034	0.332	0.026	0.410	0.029	0.395	0.045	0.400	0.034
	2/*N*	0.484	0.038	0.342	0.025	0.422	0.027	0.440	0.034	0.422	0.031
	5/*N*	0.444	0.030	0.330	0.024	0.382	0.027	0.411	0.039	0.392	0.030
1	1/*N*	0.467	0.034	0.333	0.023	0.408	0.027	0.375	0.034	0.396	0.030
	2/*N*	0.480	0.037	0.343	0.025	0.427	0.026	0.431	0.027	0.420	0.029
	5/*N*	0.446	0.035	0.329	0.021	0.384	0.025	0.407	0.034	0.392	0.029

improve them. We also observed that our designed crossover operator can indeed assist our approach in achieving better performance since the CXR of 0.2, 0.5, 0.8, and 1 outperformed the crossover rate of 0. However, there was no statistically significant difference among the performance of CXR with 0.2, 0.5, 0.8, and 1. A possible explanation could be that the optimal solutions mainly exist in the global search space rather than local search space and thus increasing the CXR cannot really help to significantly improve the performance of our approach toward finding optimal solutions. However, more experiments are required as the next step work to further investigate these interesting observations.

E. Threats to Validity

This section summarizes the identified threats that could invalidate the performed empirical evaluation. One of the *internal validity* threats lies on the parameter configurations of search algorithms (e.g., population size, number of generation, crossover rate) since different parameter settings may lead to different performances of algorithms [31]. To reduce this threat, the settings suggested in the guidelines provided by Arcuri and Briand [29] as well as the default settings of jMetal [32] were selected. Another internal validity threat could be the selection of CXR and MR. To mitigate such threat, we carefully chose the five CXR and three MR based on the guidelines from [31] and empirically evaluated in total 15 combinations. We also plan to evaluate more CXR and MR to further strengthen our observations. An *external validity* threat could be related to the generalization of the results. To deal with this issue, four independent case studies from different application domains and with different complexities were used for evaluating our approach. Moreover, one of the case studies was a real-world industrial case study. Regarding *conclusion validity* threats, one could be the random variations produced by search algorithms. To reduce this threat, the executions of the algorithms were repeated 100 times for RQ1 and RQ2 and 30 times for RQ3 and analyzed by means of statistical tests, as recommended in [29]. A *construct validity* threat might be that the measures used are not comparable across the selected algorithms. To reduce this threat, the same stopping criterion for all the algorithms was used, i.e., the number of fitness evaluations is set to 100 000 to seek the best solutions for test generation.

VI. LESSONS LEARNED

This section discusses a set of LLs extracted based on our collaboration with several industrial partners in terms of applying nature-inspired optimization in practice.

LL1: Continuously involve industrial practitioners for defining a fitness function. This LL emphasizes the importance of involving industrial practitioners for defining a fitness function when applying nature-inspired optimization techniques into practice. It is well known that a fitness function is defined to assess the solution quality during the search, and thus, defining a proper fitness function (e.g., Section IV) is highly important when applying nature-inspired methods into practice, e.g., generating TCs for testing industrial CPSs (e.g., automated warehouse systems). The first core step of defining an appropriate fitness function is to formally formulate a set of cost-effectiveness measures (see Section IV-A) based on particular industrial problems (e.g., TC generation problem). However, based on our experience, industrial practitioners (e.g., test engineers) are usually not familiar with defining complex mathematical equations for the fitness function, which is one of the core challenges when applying nature-inspired methods from research to practice. To bridge such gap, one useful means is to actively involve industrial practitioners from the beginning and let them in the loop at every step when formulating cost-effectiveness measures and a fitness function. For instance, we first discussed each objective with the test engineers of our industrial partners and formulated each objective as a cost/effectiveness measure such as requirement coverage (see Section IV-A). Afterward, we arranged several working meetings/seminars with our industrial partners and asked them to provide feedback about the defined cost/effectiveness measures. We further improved the defined measures accordingly and requested their feedback again until we (researchers and

industrial practitioners) agreed on these measures. Based on our experience, such continuous and incremental process is helpful to assist industrial practitioners in acquiring interests and good understandings of the defined measures and fitness functions. Note that the four cost/effectiveness measures defined in this work were extracted from the three industrial problems of testing CPSs, which can be considered as generic measures for testing any type of industrial CPSs. However, the defined measures can be further refined when there are particular concerns for other contexts/domains.

LL2: Be careful and comprehensive when incorporating and evaluating the fitness function with multi-objective search algorithms. Once the fitness function is defined, the next step is to incorporate it with multi-objective search algorithms (e.g., NSGA-II) followed by running the search. Based on our experience, in total four points were extracted that should be paid attention for this step. First, multi-objective search algorithms usually have intrinsic randomness due to, for example, initial random population, crossover, and mutation operators. Thus, it is suggested to run each algorithm at least ten times, to reduce the random variations produced by the algorithms. Second, the state-of-the-art has shown that different problems may lead to totally different performances of multi-objective search algorithms [30]. Thus, a comprehensive algorithm comparison should be performed with the aim to obtaining the best one. For instance, in this paper, a total of five multi-objective search algorithms are selected for assessing the algorithm performance along with our defined fitness function. Note that we also selected RS for a sanity check. Third, it is recommended to apply standard statistical tests to analyze the results produced by the search and extract the key findings. For instance, we used the Mann–Whitney U test to determine the significance of the results obtained by different algorithms. Fourth, multi-objective search algorithms usually return a set of nondominated solutions with the equivalent quality, and thus, industrial practitioners can select one solution at random to use when there is no any specific concerns. However, when there are one or more objectives that are more important than the others (e.g., RC), we would recommend practitioners to choose appropriate solutions by considering the corresponding objective, e.g., selecting a solution with the highest value of RC.

LL3: Remember tool support with user friendliness is crucial for success adaptions of nature-inspired methods in practice. Researchers usually talk about breakthrough ideas but industrial practitioners are more concerned about tools. This LL underlines the importance of a user-friendly tool support when applying nature-inspired methods into the industry. We suggest researchers pushing themselves hard by maturing their research approaches as user-friendly tool support, which can be eventually applied by industrial practitioners. In our case, we designed and developed a tool chain on the top of several existing tools, e.g., FeatureIDE and MATLAB Simulink based on the discussions with our industrial partners.

VII. Conclusion and Future Work

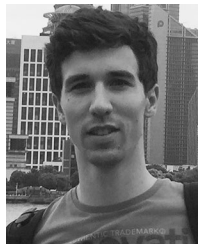
This paper proposes a search-based multi-objective approach for systematically generating and prioritizing cost-effective reactive TCs for testing CPSs. To this end, we defined a fitness function with four objectives to guide the search toward finding optimal solutions. We also designed one crossover operator and two mutation operators. The fitness function and designed operators were integrated with five different search algorithms and evaluated using four case studies. The results showed that NSGA-II together with our fitness function and operators demonstrated the best performance and managed to outperform the rest of the algorithms on average between 11.93% and 47.07% in terms of the HV quality indicator. Moreover, the crossover rate of 0.2 and the mutation rate of $2/N$ along with our approach is recommended in order to achieve the best performance.

In the future, we plan to extend our work from four perspectives. More specifically, we first plan to include more industrial case studies to further strengthen our approach. We also plan to involve industrial practitioners for deploying and thoroughly testing our approach in their current practice. Second, we plan to refine our fitness function by including more objectives (cost/effectiveness measures). For instance, we want to take pairwise coverage of system functionalities into account since many real faults of systems are caused by the interactions of different system functionalities. Third, we would like to investigate a formal approach to link requirements with reactive behavior of the system. This would allow to automatically generate the script function that measures RC, leading our test generation approach to be more systematic. Fourth, we would like to tune the performance of our approach with more parameters, e.g., different population sizes, number of generations. We also plan to evaluate the performance of our approach with other multi-objective search algorithms, such as archive-based hybrid scatter search, which combines both scatter search and genetic algorithms.

References

[1] P. Derler, E. A. Lee, and A. Sangiovanni-Vincentelli, "Modeling cyber-physical systems," *Proc. IEEE*, vol. 100, no. 1, pp. 13–28, Jan. 2011.
[2] P. J. Mosterman and J. Zander, "Industry 4.0 as a cyber-physical system study," *Softw. Syst. Model.*, vol. 15, no. 1, pp. 17–29, 2016.
[3] Y. Jiang, H. Song, R. Wang, M. Gu, J. Sun, and L. Sha, "Data-centered runtime verification of wireless medical cyber-physical system," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 1900–1909, Aug. 2017.
[4] P. J. Mosterman and J. Zander, "Cyber-physical systems challenges: A needs analysis for collaborating embedded software systems," *Softw. Syst. Model.*, vol. 15, pp. 5–16, 2016.
[5] P. Leitão, A. W. Colombo, and S. Karnouskos, "Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges," *Comput. Ind.*, vol. 81, pp. 11–25, 2016.
[6] A. Fisher, C. A. Jacobson, E. A. Lee, R. M. Murray, A. Sangiovanni-Vincentelli, and E. Scholte, "Industrial cyber-physical systems—ICyPhy," in *Complex Systems Design & Management*. New York, NY, USA: Springer, 2014, pp. 21–37.
[7] L. Briand, S. Nejati, M. Sabetzadeh, and D. Bianculli, "Testing the untestable: Model testing of complex software-intensive systems," in *Proc. 38th Int. Conf. Softw. Eng. Companion*, 2016, pp. 789–792.
[8] R. Matinnejad, S. Nejati, L. C. Briand, and T. Bruckmann, "Automated test suite generation for time-continuous simulink models," in *Proc. 38th Int. Conf. Softw. Eng.*, 2016, pp. 595–606.
[9] A. Arrieta, G. Sagardui, L. Etxeberria, and J. Zander, "Automatic generation of test system instances for configurable cyber-physical systems," *Softw. Qual. J.*, vol. 25, no. 3, pp. 1041–1083, 2017.
[10] A. Arrieta, S. Wang, G. Sagardui, and L. Etxeberria, "Test case prioritization of configurable cyber-physical systems with weight-based search algorithms," in *Proc. 2016 Genet. Evol. Comput. Conf.*, 2016, pp. 1053–1060.

[11] R. Feldt, S. M. Poulding, D. Clark, and S. Yoo, "Test set diameter: Quantifying the diversity of sets of test cases," in *Proc. 2016 IEEE Int. Conf. Softw. Test., Verification Validation*, Apr. 2016, pp. 223–233.
[12] A. Arrieta, S. Wang, G. Sagardui, and L. Etxeberria, "Search-based test case selection of cyber-physical system product lines for simulation-based validation," in *Proc. 20th Int. Syst. Softw. Product Line Conf.*, 2016, pp. 297–306.
[13] A. Arrieta, S. Wang, U. Markiegi, G. Sagardui, and L. Etxeberria, "Search-based test case generation for cyber-physical systems," in *2017 IEEE Congr. Evol. Comput.*, 2017, pp. 688–697.
[14] P. J. Mosterman, D. E. Sanabria, E. Bilgin, K. Zhang, and J. Zander, "Automating humanitarian missions with a heterogeneous fleet of vehicles," *Annu. Rev. Control*, vol. 38, no. 2, pp. 259–270, 2014.
[15] W. Mueller, M. Becker, A. Elfeky, and A. DiPasquale, "Virtual prototyping of cyber-physical systems," in *Proc. Asia South Pac. Des. Autom. Conf.*, 2012, pp. 219–226.
[16] H. Shokry and M. Hinchey, "Model-based verification of embedded software," *Computer*, vol. 42, no. 4, pp. 53–59, 2009.
[17] J. Zander-Nowicka, "Reactive testing and test control of hybrid embedded software," in *Proc. 5th Workshop Syst. Test. Validation*, 2007, pp. 45–62.
[18] M. Mohaqeqi, M. R. Mousavi, and W. Taha, "Conformance testing of cyber-physical systems: A comparative study," *ECEASST*, vol. 70, pp. 1–16, 2014. [Online]. Available: http://journal.ub.tu-berlin.de/eceasst/article/view/982
[19] M. Mohaqeqi and M. R. Mousavi, "Sound test-suites for cyber-physical systems," in *Proc. 2016 10th Int. Symp. Theor. Aspects Softw. Eng.*, 2016, pp. 42–48.
[20] H. Abbas, B. Hoxha, G. Fainekos, J. V. Deshmukh, J. Kapinski, and K. Ueda, "Wip abstract: Conformance testing as falsification for cyber-physical systems," in *Proc. 2014 ACM/IEEE Int. Conf. Cyber-Phys. Syst.*, 2014, pp. 211–211.
[21] A. Aerts, M. A. Reniers, and M. R. Mousavi, "Model-based testing of cyber-physical systems," in *Cyber-Physical Systems: Foundations, Principles and Applications*. New York, NY, USA: Academic, 2016, pp. 287–304.
[22] M. Woehrle, K. Lampka, and L. Thiele, "Conformance testing for cyber-physical systems," *ACM Trans. Embedded Comput. Syst.*, vol. 11, no. 4, pp. 84:1–84:23, Jan. 2013. [Online]. Available: http://doi.acm.org/10.1145/2362336.2362351
[23] R. Ben Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, "Testing advanced driver assistance systems using multi-objective search and neural networks," in *Proc. 31st IEEE/ACM Int. Conf. Autom. Softw. Eng.*, 2016, pp. 63–74.
[24] T. E. J. Vos *et al.*, "Evolutionary functional black-box testing in an industrial setting," *Softw. Qual. J.*, vol. 21, no. 2, pp. 259–288, 2013.
[25] J. Zander-Nowicka, "Model-based testing of real-time embedded systems in the automotive domain," Ph.D. dissertation, Tech. Univ. Berlin, Berlin, Germany, 2008.
[26] E. Lehmann, "Time partition testing: A method for testing dynamic functional behaviour," in *Proc. Eur. Softw. Test Congr.*, 2000, pp. 1–11.
[27] H. Hemmati, A. Arcuri, and L. Briand, "Achieving scalable model-based testing through test case diversity," *ACM Trans. Softw. Eng. Methodol.*, vol. 22, no. 1, pp. 6:1–6:42, 2013.
[28] T. Thuem, C. Kastner, F. Benduhn, J. Meinicke, G. Saake, and T. Leich, "Featureide: An extensible framework for feature-oriented software development," *Sci. Comput. Program.*, vol. 79, pp. 70–85, 2014.
[29] A. Arcuri and L. Briand, "A practical guide for using statistical tests to assess randomized algorithms in software engineering," in *Proc. 2011 33rd Int. Conf. Softw. Eng.*, 2011, pp. 1–10.
[30] S. Wang, S. Ali, T. Yue, Y. Li, and M. Liaaen, "A practical guide to select quality indicators for assessing pareto-based search algorithms in search-based software engineering," in *Proc. 38th Int. Conf. Softw. Eng.*, 2016, pp. 631–642.
[31] A. Arcuri and G. Fraser, "Parameter tuning or default values? An empirical investigation in search-based software engineering," *Empirical Softw. Eng.*, vol. 18, no. 3, pp. 594–623, 2013.
[32] J. J. Durillo and A. J. Nebro, "jMetal: A java framework for multi-objective optimization," *Adv. Eng. Softw.*, vol. 42, no. 10, pp. 760–771, 2011.
[33] A. Vargha and H. D. Delaney, "A critique and improvement of the CL common language effect size statistics of McGraw and Wong," *J. Educ. Behav. Statist.*, vol. 25, no. 2, pp. 101–132, 2000.



Aitor Arrieta received the B.Sc. degree in industrial electronics engineering from the University of Mondragon, Mondragon, Spain, in 2012, the B.Sc. degree with a major in automation engineering from the University of Skövde, Skövde, Sweden, in 2012, the Master's degree in embedded systems from the University of Mondragon, in 2014, and the Ph.D. degree (with honors) in computer science from the University of Mondragon, in 2017.

He is a Lecturer and a Researcher with the Embedded Systems Group, Electronics and Computer Science Department, University of Mondragon. His main research interests include cyber-physical systems, embedded systems, test and validation methodologies, and product line engineering.

Shuai Wang received the Ph.D. degree (with honors) in computer science from the University of Oslo, Oslo, Norway, in 2015.

He is currently a Postdoctoral Researcher with Simula Research Laboratory, Fornebu, Norway. His research interests include search-based software engineering, product line engineering, model-based testing, and empirical software engineering with more than 30 publications from well-recognized international journals (such as the IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the *Journal of Systems and Software*, and the *Journal of Empirical Software Engineering*) and highly reputed international conferences (such as ICSE, ICSME, MODELS, ISSRE, ICST, and SPLC).

Dr. Wang is the recipient of the ACM Distinguished Paper Award of MODELS 2013 (also best application track paper), two Best Paper Nominees (SPLC 2014 and ICST 2017), and the Outstanding Reviewer Award during 2015–2016 of the *Information and Software Technology* journal. He is a Member of the ACM and IEEE Computer Society.

Urtzi Markiegi received the B.Sc. degree in computer science engineering in 2013 from the University of Mondragon, Mondragon, Spain, where he is currently working toward Ph.D. degree at the Electronics and Computer Science Department.

He is a Lecturer and a Researcher with the Electronics and Computer Science Department, University of Mondragon. His research interests include software engineering, testing, validation, and product line engineering.

Goiuria Sagardui received the Ph.D. degree in computer science from the University of the Basque Country, Leioa, Spain, in 1999.

She is the Head of the Embedded Systems Group and Software Engineering Group, University of Mondragon, Mondragon, Spain. Her research interest focuses on software engineering, including software product lines and model-driven engineering.

Leire Etxeberria received the B.Sc. degree in computer science engineering and the Ph.D. degree in computer science from Mondragon University, Mondragon, Spain, in 2004 and 2008, respectively.

She is a Lecturer and a Researcher with the Electronics and Computer Science Department, University of Mondragon. Her research interests include software product lines, model-driven development, variability and V&V, variability and safety, etc., in the embedded systems domain.