Search-Based Selection and Prioritization of Test Scenarios for Autonomous Driving Systems

Chengjie Lu[1(✉)], Huihui Zhang[2], Tao Yue[1,3], and Shaukat Ali[3]

[1] Nanjing University of Aeronautics and Astronautics, Nanjing, China
chengjielu@nuaa.edu.cn, taoyue@ieee.org
[2] Weifang University, Weifang, China
huihui@wfu.edu.cn
[3] Simula Research Laboratory, Oslo, Norway
shaukat@simula.no

Abstract. Violating the safety of autonomous driving systems (ADSs) could lead to fatal accidents. ADSs are complex, constantly-evolving and software-intensive systems. Testing an individual ADS is challenging and expensive on its own, and consequently testing its multiple versions (due to evolution) becomes much more costly. Thus, it is needed to develop approaches for selecting and prioritizing tests for newer versions of ADSs based on historical test execution data of their previous versions. To this end, we propose a multi-objective search-based approach for Selection and Prioritization of tEst sCenarios for auTonomous dRiving systEms (SPECTRE) to test newer versions of an ADS based on four optimization objectives, e.g., demand of a test scenario put on an ADS. We experimented with five commonly used multi-objective evolutionary algorithms and used a repository of 60,000 test scenarios. Among all the algorithms, IBEA achieved the best performance for solving all the optimization problems of varying complexity.

Keywords: Test optimization · Multi-objective search · Autonomous driving

1 Introduction

Autonomous Driving Systems (ADSs) are safety-critical systems, thus requiring a high degree of dependability. Testing ADSs provides confidence that such systems are dependable; however, due to their complex implementation and the mandate to deal with complex environment, testing ADSs is challenging [3]. Moreover, in practice, ADSs evolve, e.g., because of introducing a new functionality and updating an existing one. Thus, testing ADSs, in general, is costly, especially considering that their new versions need to be tested continuously. Thus, it is important to optimize tests for ADSs. Motivated by this, we propose a search-based approach for Selection and Prioritization of tEst sCenarios for

U.-M. O'Reilly and X. Devroey (Eds.): SSBSE 2021, LNCS 12914, pp. 41–55, 2021.
https://doi.org/10.1007/978-3-030-88106-1_4

auTonomous dRiving systEms (SPECTRE) to test a new version of an ADS from existing test scenarios designed for its previous versions.

Figure 1 shows SPECTRE's overview. SPECTRE relies on test scenario execution results from a previous version of an ADS to prioritize test scenarios to be executed on its newer version. In contrast, the existing works on Search-based Testing (SBT) of ADSs (e.g. [4,5,10]) focus on generating test scenarios for testing ADSs.

In SPECTRE, each scenario is characterized with a set of properties of the ego vehicle with the ADS under test deployed (e.g., acceleration, speed) and its environment (e.g., weather, number of obstacles). The simulation of each test scenario leads to output four key values (*Execution Results* in Fig. 1): (1) whether a collision occurred with the scenario, (2) collision probability associated with the scenario, (3) the extent of demand on the ADS put by the scenario, and (4) diversity of the scenario as compared to the others. Based on these attributes, we define four optimization objectives.

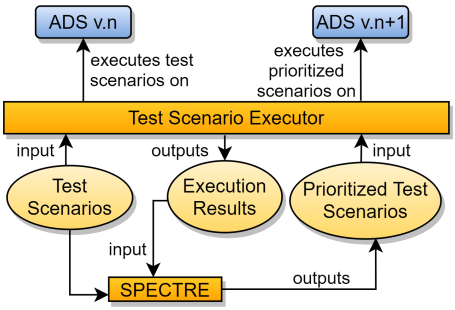

Fig. 1. SPECTRE – overview

To solve our optimization problem, we implemented SPECTRE with five Multi-Objective Evolutionary Algorithms (MOEAs): NSGA-II, NSGA-III, IBEA, SPEA2 and MOCell. Random Search (RS) was used for sanity check. To evaluate SPECTRE, we employed a repository with 60,000 test scenarios and their execution results, Baidu Apollo[1] as the software under test, together with the LGSVL simulator[2]. Results showed that IBEA performed the best in terms of producing quality solutions for all the optimization problems of varying complexity.

The paper is organized as follows: Sect. 2 formulates the search problem and Sect. 3 presents the evaluation, followed by experiment results in Sect. 4. Section 5 is the related work and Sect. 6 concludes the paper.

2 Problem Representation and Objective Function

2.1 Problem Representation

Given a set of test scenarios $SS = \{S_1, S_2 ... S_{ns}\}$, with ns being the total size of SS, SPECTRE selects a subset of test scenarios from SS and prioritize them to construct a new test suite to test a new version of the ADS. A scenario $S \in SS$ is characterized with a list of properties (e.g., speed of the ego vehicle, weather of the environment): $\{p_1, p_2, p_3 ... p_{np}\}$, where np is the total number of properties.

[1] https://apollo.auto/.
[2] https://www.svlsimulator.com/.

Basic Concepts. Based on historical execution results of test scenarios, in our context, each scenario S is attached with values of the attributes below. Notice that values of these attributes are extracted from execution results of test scenarios, which are different from properties characterizing test scenarios.

Attribute-1 (Collision (COL)) is a Boolean attribute telling if a scenario S led the ADS collide with obstacles. With COL, we classify all scenarios into either *Collision Scenarios* (S_{COL}) or *Non-Collision Scenarios* (S_{NCOL}).

Attribute-2 (Collision Probability (CPT) measures how close the ADS is to collide with obstacles in a scenario S. We use current distance (CD) and safety distance (SD) [7] to measure CPT as:

$$CPT = \begin{cases} \dfrac{SD - CD}{SD}, & CD < SD \\ 0.0, & else \end{cases} \tag{1}$$

where, SD is a function of accelerations and velocities of the ego vehicle and an obstacle: $F_{SD}(a_{ego}, a_{obstacle}, v_{ego}, v_{obstacle})$; CD is a function of positions of the ego vehicle and an obstacle: $F_{CD}(Pos_{ego}, Pos_{obstacle})$. Notice that, based on Equation (1), the greater the degree of the current distance violation, the higher the collision probability. Based on CPT, a scenario S is defined as *Potential Collision Scenario* (S_{PCOL}) if $CPT \in (0, 1)$, implying that when driving in S with $CPT \in (0, 1)$, the ego vehicle has a chance to collide with obstacles.

Attribute-3 (Demand (DEM)). Inspired by [8], we use the concept of *demand* to measure how much difficulty the generated scenarios put the ego vehicle in, based on their *np* properties. Concretely, each scenario property (e.g., the speed) has a corresponding *demand* value. For example, considering that when facing the same or a similar environment, a higher speed of the ego vehicle would possibly result in a higher *demand*. For instance, the speed ranges from 0 to 80 km per hour when driving on an urban road, and its demand can be classified into four categories (based on the level of risk that the vehicle will drive from SFMTA[3]): 0, 1, 3, and 4 for the *Zero*, *Slow*, *Moderate*, and *Fast* speed, respectively. Similarly, demand values for property *rain* can be: 0, 1, 2, and 3, representing no rain, light, moderate or heavy rain, respectively.

Furthermore, we consider test scenarios with demand values of one of its properties equal or greater than the medium value, i.e., $(Max_D - Min_D)/2$ as high demand values, where, Max_D and Min_D denote the maximum and minimum demands of one property. For instance, for property *rain* (i.e., medium $= (3 - 0)/2$), *Moderate* (2) or *Heavy* (3) rain can be considered as high demand. When combining the *np* properties of a scenario together, its DEM is defined as the number of high *demand* properties, taking an integer values in [0, np]. Based on DEM, a scenario is defined as *High Demand Scenario* (S_{HighD}) if more than half of its properties are high demand properties.

Attribute-4 (Diversity (DIV)). Considering that values of different properties (e.g., speed and throttle) of a scenario S are not comparable, we first apply the normalization function [11] below to ensure all the values fall into [0, 1].

[3] https://www.sfmta.com.

$$nor(F(x)) = \frac{F(x) - F_{min}}{F_{max} - F_{min}} \tag{2}$$

Based on the definition of scenarios, first, we define the diversity of the k_{th} property in two scenarios S_i, S_j as below:

$$PDIV_k = nor(|p_{ik} - p_{jk}|), \tag{3}$$

where, p_{ik} and p_{jk} are the k_{th} property values of S_i and S_j, respectively. Then we compute the scenario diversity $SDIV$ of two scenarios S_i, S_j as below:

$$SDIV_{i,j} = \frac{\sum_{k=0}^{np} PDIV_k}{np}. \tag{4}$$

where np is the number of property used to define scenarios. DIV of a scenario S_i is then calculated as:

$$DIV = \sum_{j=0}^{ns} SDIV_{i,j}, j \neq i. \tag{5}$$

where ns is the total number of scenarios in the test scenario set SS. Let test suite TS be a set of prioritized test scenarios $TS = \{S_1, S_2, S_3...S_{nts}\}$ selected from SS, where nts is the size TS, a search budget given by users. $PS = \{TS_1, TS_2, TS_3...TS_{nps}\}$ (with nps being its size) is the entire search space of all possible solutions, i.e., the set of permutations of the ordering of all the scenarios from SS. So, if we want to select and prioritize nts scenarios, the size of the search space nps is: $nps = A_{ns}^{nts} = ns*(ns-1)*(ns-2)*...*(ns-nts+1)$. The search space nps will exponentially increase with the growth of ns and exhaustively exploring the entire search space is practically infeasible. Thus, search algorithms can be applied to find optimal solutions within a given budget.

Given $TS = \{S_1, S_2, S_3...S_{nts}\}$, a prioritization solution $X = \{x_1, x_2, ..., x_{nts}\}$ is a particular permutation of TS, where x_i $(0 \leq x_i \leq nts - 1)$ denotes the unique position of test scenario S_i. Note that the value of x_i is unique and ranges from 0 to $nts - 1$. If $x_i = \mathrm{j}$, S_i is the $(j+1)_{th}$ scenario in the sequence. A smaller value of x_i means a preceding position in the sequence X, that is, 0 means the first position while $nts - 1$ indicates the last position of a test scenario.

Optimization Problem. Given a set of test scenarios $SS = \{S_1, S_2...S_{ns}\}$ and a desired budget nts $(nts \leq ns)$, find solution $TS_k \in PS$ with a particular permutation $X_{kp} = \{x_1^{kp}, x_2^{kp}, x_3^{kp}...x_{nts}^{kp}\}$, where TS_k includes nts scenarios selected and prioritized from SS that satisfy:

(1) $\forall TS_i \in PS : \#S_{COL}^{TS_i} \leq \#S_{COL}^{TS_k}$, telling that TS_k has the maximum number of collision scenarios.
(2) $\forall TS_i \in PS : \#S_{PCOL}^{TS_i} \leq \#S_{PCOL}^{TS_k}$, implying that TS_k has the maximum number of potential collision scenarios.
(3) $\forall TS_i \in PS : \#S_{HighD}^{TS_i} \leq \#S_{HighD}^{TS_k}$, which denotes that TS_k has the maximum number of highly demand scenarios.

(4) $\forall TS_i \in PS : DIV(TS_k) \geq DIV(TS_j)$, which indicates that TS_k has the most diverse scenarios.

(5) $\forall(x_i < x_j(x_i, x_j \in X_{kp} \wedge i \neq j)) : \begin{cases} COL^{X_{kp}}(x_j) \leq COL^{X_k}(x_i) \wedge \\ CPT^{X_{kp}}(x_j) \leq CPT^{X_k}(x_i) \wedge \\ DEM^{X_{kp}}(x_j) \leq DEM^{X_k}(x_i) \wedge \\ DIV^{X_{kp}}(x_j) \leq DIV^{X_k}(x_i) \end{cases}$ which means values of the four attributes of the scenarios in TS_k will descend.

2.2 Objective Functions

Based on the definitions in Sects. 2.1, we formally define the objectives.

Objectives 1 to 4 are for selecting NTS test scenarios that maximize these objectives, whereas objective 5 is devised to prioritize the selected test scenarios.

Objective 1. ***Number of Collision Scenarios*** (NS_{COL}) is the number of *Collision Scenarios* S_{COL} in TS_k which led the ego vehicle to collide.

$$NS_{COL}^{TS_k} = Count(S_{COL}), \tag{6}$$

Objective 2. ***Number of Potential Collision Scenarios*** (NS_{PCOL}) counts the number of *Potential Collision Scenarios* (S_{PCOL}) (Sect. 2.1):

$$NS_{PCOL}^{TS_k} = Count(S_{PCOL}) \tag{7}$$

Objective 3. ***Number of High Demand Scenarios*** (NS_{HighD}) is the number of *High Demand Scenarios* S_{HighD} defined in Sect. 2.1:

$$NS_{HighD}^{TS_k} = Count(S_{HighD}) \tag{8}$$

Objective 4. ***Test Solution Diversity*** ($TSDIV$) measures the differences of each pair of scenarios (calculated with Eq. 4) in TS_k as:

$$TSDIV_{TS_k} = \frac{\sum_{i=0}^{nts-1} \sum_{j=i+1}^{nts} SDIV_{i,j}}{nts} \tag{9}$$

Objective 5. Attribute Prioritization ($APrio$) differentiates scenarios based on positions of attribute values. In Eq. 11, $\frac{nts-x_i}{nts}$ indicates that a value of an attribute (e.g., COL) in a preceding position has a higher contribution to $APrio$. When putting equal weights to each attribute, we can obtain one value: $APrio_{TS_k}$ (Eq. 10), telling that the objective function tries to permute scenarios with higher collision, collision probability, demand and diversity in the front positions of the permutation sequence as early as possible.

$$APrio_{TS_k} = \omega 1 * APrio_{col} + \omega 2 * APrio_{cpt} + \omega 3 * APrio_{dem} + \omega 4 * APrio_{div}; \quad (\omega 1 = \omega 2 = \omega 3 = \omega 4 = 0.25) \tag{10}$$

$$APrio_{col} = \frac{\sum_{i=1}^{nts} * \frac{nts - x_i}{nts} * COL_i}{nts}; APrio_{cpt} = \frac{\sum_{i=1}^{nts} * \frac{nts - x_i}{nts} * CPT_i}{nts};$$
$$APrio_{dem} = \frac{\sum_{i=1}^{nts} * \frac{nts - x_i}{nts} * DEM_i}{nts}; APrio_{div} = \frac{\sum_{i=1}^{nts} * \frac{nts - x_i}{nts} * DIV_i}{nts} \quad (11)$$

3 Empirical Evaluation

3.1 DataSet

Test Scenarios. A test scenario for ADSs, in our experiment, is defined with 19 properties; 5 of them are about the ego vehicle (e.g., acceleration and speed) and the other 14 properties describe the environment (e.g., pedestrians, weather). To produce the dataset, we employed the Baidu Apollo Open Platform 5.0 as the ADS under test and integrated it with the LGSVL simulator. We chose the San Francisco map for scenarios collecting because it has a large number of different types of roads such as one-way roads, two-way roads and cross walks.

Collecting Data. The test scenarios in the dataset were automatically collected when we were testing Apollo 5.0 with a machine learning based environment configuration generation strategy. We executed the strategy, together with LGSVL for nearly 1000 times on four different roads of the San Francisco map loaded in LGSVL. In the end, we managed to collect the dataset of 90K test scenarios, each of which is characterized with the 19 properties.

Processing Data. First, we removed duplicated test scenarios from the original dataset, to reduce unnecessary effort for prioritization In the end, we obtained a dataset containing 60K scenarios. To save time on calculating diversity during the execution of the MOEAs, we further calculated all the pair-wise comparison diversity values of the test scenarios in advance, i.e., *SDIV*.

Labelling Test Scenarios with Their Attributes. After data collecting and pre-processing, for each test scenario, we calculated four values for each of the four attributes (Sect. 2.1). The labeled dataset was then fed to SPECTRE for selection and prioritization. The replication package is available at Github[4].

3.2 Research Questions (RQs)

RQ1: How do the selected MOEAs compare to each other in terms of solving our optimization problem?

RQ2: How the selected MOEAs compare to each other when solving optimization problems of various search budgets?

RQ3: How does the search budget affect the effectiveness of the selected MOEAs?

RQ4: How is the time performance of the selected MOEAs?

[4] https://github.com/ssbse2021/SPECTRE.

3.3 Experiment Design and Evaluation Metrics

To answer RQs, we integrated SPECTRE with five commonly used MOEAs: NSGA-II, NSGA-III, IBEA, SPEA2 and MOCell for solving different Search-Based Software Engineering (SBSE) problems (e.g., test selection [19], test minimization [21], requirements prioritization [20]). Among them, NSGA-II and SPEA2 were chosen because a large number of works studied them and have proven to be effective for solving SBSE problems [16]; MOCell is an effective Pareto-based MOEA used in [16]; NSGA-III aims to handle many-objective (i.e., 3 to 15) optimization problems; and IBEA represents Indicator-based evolutionary algorithms. For IBEA, we employed the I_{ϵ^+}-indicator [22]. As recommended in [2], RS was used for sanity check. We used the default parameter settings of MOEAs from jMetal [9], except for NSGA-III's population size, which was changed to 100 from the default size of 92 to be consistent with the other MOEAs. All parameter settings are presented in SPECTRE's online repository.

SPECTRE is configured to select test scenarios with eight NTS settings, where NTS is the number of test scenarios to be selected from the dataset, ranging from 1,000 to 8,000 with an increment of 1,000. We executed SPECTRE for 30 times for each MOEA and with each NTS.

Based on the published guide [1], we used the Inverted Generational Distance (IGD) quality indicator. IGD computes the distance of the solutions of the Pareto front from those of the reference Pareto front to assess the performance of the MOEAs. Thus, a smaller IGD value indicates a better performance. For each NTS, a reference Pareto front is computed by merging all the Pareto fronts from all the 30 runs of all the MOEAs and RS.

3.4 Statistical Tests

Based on the guidelines in [2], we selected statistical tests with the significance level of 0.05. First, we applied the Kolmogorov-Smirnov test to test if the samples are normally distributed. Results tell that they are not normally distributed. Then, we used the non-parametric Kruskal-Wallis rank test to compare the samples. Results reveal a significant difference among them. Thus, we performed the Mann-Whitney U test for pairwise comparisons, and used the Vargha and Delaney effect size to calculate $\hat{A}_{12}$. Given metric χ, $\hat{A}_{12}$ was used to compare the probability of yielding higher values of χ for algorithms AlgA and AlgB. If $\hat{A}_{12}$ is 0.5, then the results were obtained by chance. If $\hat{A}_{12}$ is greater than 0.5, then AlgA has a higher chance to achieve a better performance than AlgB in terms of χ, and vice versa. The Mann-Whitney U test returns p-values to check if the difference between AlgA and AlgB is significant. We also adjusted all p-values with the Bonferroni Correction to avoid type **I** errors.

To study the correlation between IGD and NTS, we performed the Spearman's rank correlation (ρ) test. The ρ value ranges from -1.0 to 1.0, i.e., there is a positive correlation if ρ is close to 1.0 and a negative correlation when ρ closes to -1.0. If ρ closes to 0 then there is no correlation between IGD and NTS. We also reported the significance of the correlation using p-value, i.e., a p-value less than 0.05 tells that the correlation is statistically significant.

4 Results and Analyses

To check if the problem is complex, we compared the selected MOEAs with RS. Results show that all the MOEAs performed significantly better than RS in terms of IGD for all the NTSs. Thus, for answering our RQs, we didn't include RS. Detailed results can be found in SPECTRE's online repository.

4.1 Results of RQ1

Results for RQ1, comparing each MOEA with the others in terms of p-values and $\hat{A}_{12}$, are reported in Table 1. For IBEA, when comparing with the other MOEAs, all the p-values are less than 0.05 and $\hat{A}_{12} < 0.5$, thus suggesting that IBEA is significantly better than the other MOEAs. NSGA-II is significantly better than MOCell, but significantly worst than NSGA-III, SPEA2, and IBEA. NSGA-III is significantly better than MOCell and NSGA-II, whereas significantly worst than IBEA. MOCell was significantly worst than the rest. We can then rank the MOEAs as: IBEA, NSGA-III/SPEA2, NSGA-II, and MOCell.

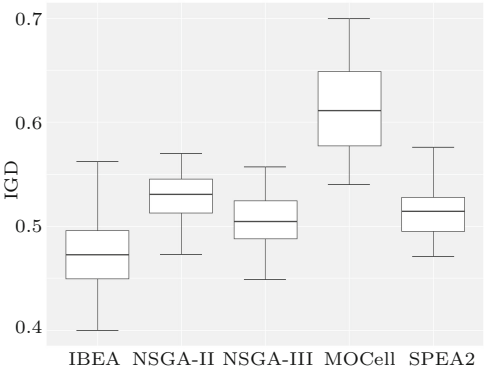

Fig. 2. Descriptive statistics of IGD when combining all NTS results – RQ1

Figure 2 better illustrates the results, especially the variance of 30 IGD values produced by each MOEA in its 30 runs. From the figure, we see that MOCell performed the worst and also produced results with the largest variance. The variances of IGD values of the other four MOEAs are smaller and comparable.

We therefore recommend using IBEA for solving our search problem.

Table 1. Results for comparing MOEAs when combining all NTS results – RQ1

Metric	IBEA *vs.*				NSGA-II *vs.*			NSGA-III *vs.*		MOCell *vs.*
	NSGA-II	NSGA-III	MOCell	SPEA2	NSGA-III	MOCell	SPEA2	MOCell	SPEA2	SPEA2
$\hat{A}_{12}$	**0.121**	**0.287**	**0.008**	**0.207**	0.754	**0.044**	0.678	**0.009**	0.399	0.973
p-value	<**0.05**	<**0.05**	<**0.05**	<**0.05**	<0.05	<**0.05**	<0.05	<**0.05**	0.183	<0.05

*A bold $\hat{A}_{12} < 0.5$ with a $p - value < 0.05$ implies that the upper algorithm is significantly better than the bottom one, whereas $\hat{A}_{12} > 0.5$ with $p - value < 0.05$ means vice versa. A $p - value > 0.5$ means no significant differences.

4.2 Results of RQ2

Table 2 reports pair-wise comparisons of MOEAs with respect to IGD. Among all the MOEAs, MOCell achieved the worst performance for all NTSs. For $NTS \in \{1000, 3000\}$: IBEA achieved the best performance, followed by NSGA-III; NSGA-III significantly outperformed SPEA2 and SPEA2 was significantly better than NSGA-II. For $NTS \in \{2000, 4000, 6000, 7000\}$, IBEA significantly

outperformed the others and NSGA-III achieved the second best. No significant difference was observed between SPEA2 and NSGA-II. For $NTS = 5000$, IBEA performed the best. Both NSGA-III and SPEA2 were significantly better than NSGA-II and there was no significant difference between NSGA-III and SPEA2. For $NTS = 8000$, both IBEA and NSGA-III achieved the best, and there was no significant difference between them, and no significance difference was observed between SPEA2 and NSGA-II. Table 3 better organizes the results of the pair-wise comparisons of the MOEAs for each NTS.

Table 2. Results of comparing MOEAs for each NTS – RQ2

NTS	IBEA *vs.*				NSGA-II *vs.*			NSGA-III *vs.*		MOCell *vs.*
	NSGA-II	NSGA-III	MOCell	SPEA2	NSGA-III	MOCell	SPEA2	MOCell	SPEA2	SPEA2
1000	**.027/<.05**	**.249/<.05**	**.009/<.05**	**.098/<.05**	.834/<.05	**.032/<.05**	.653/<.05	**.018/<.05**	**.289/<.05**	.972/<.05
2000	**.040/<.05**	**.239/<.05**	**.013/<.05**	**.052/<.05**	.826/<.05	**.121/<.05**	.629/.08	**.039/<.05**	**.227/<.05**	.922/<.05
3000	**.044/<.05**	**.258/<.05**	**.007/<.05**	**.062/<.05**	.904/<.05	**.128/<.05**	.707/<.05	**.016/<.05**	**.182/<.05**	.943/<.05
4000	**.061/<.05**	**.271/<.05**	**.029/<.05**	**.096/<.05**	.851/<.05	**.177/<.05**	.578/.304	**.071/<.05**	**.210/<.05**	.858/<.05
5000	**.068/<.05**	**.206/<.05**	**.002/<.05**	**.146/<.05**	.770/<.05	**.036/<.05**	.707/<.05	**.009/<.05**	.410/.234	.984/<.05
6000	**.038/<.05**	**.258/<.05**	**.014/<.05**	**.088/<.05**	.826/<.05	**.116/<.05**	.599/.191	**.046/<.05**	**.267/<.05**	.898/<.05
7000	**.100/<.05**	**.257/<.05**	**.004/<.05**	**.116/<.05**	.769/<.05	**.044/<.05**	.559/.438	**.023/<.05**	**.264/<.05**	.963/<.05
8000	**.144/<.05**	.359/.061	**.001/<.05**	**.204/<.05**	.808/<.05	**.060/<.05**	.636/.072	**.003/<.05**	**.289/<.05**	.976/<.05

*The value before/is $\hat{A}_{12}$ and after is p-value. A bold $\hat{A}_{12} < 0.5$ with a $p - value < 0.05$ implies that the upper algorithm is significantly better than the bottom one, whereas $\hat{A}_{12} > 0.5$ with $p - value < 0.05$ means vice versa. A $p - value > 0.5$ means no significant differences.

Table 3. Ranking of MOEAs for each NTS value – RQ2

NTS	Ranking	NTS	Ranking	NTS	Ranking	NTS	Ranking
1000	I, N-III, S, N-II, M	2000	I, N-III, S/N-II, M	3000	I, N-III, S, N-II, M	4000	I, N-III, S/N-II, M
5000	I, N-III/S, N-II, M	6000	I, N-III, S/N-II, M	7000	I, N-III, S/N-II, M	8000	I/N-III, S/N-II, M

*I: IBEA, N: NSGA; S: SPEA2. M: MOCell; a/means two MOEAs have the same ranking.

The variance of 30 IGD values for each MOEA corresponding to its 30 runs for each NTS value are reported in Fig. 3. For almost all the NTSs, MOCell is the worst with the large variances. For the other MOEAs, we can observe smaller variances and they are comparable for most of NTS values. This observation is consistent with the results we obtained for RQ1.

4.3 Results of RQ3

RQ3 aims to study how the increasing number of NTSs affects the performance of each MOEA. To this end, we plot the average of 30 IGD values for NTS in Fig. 4. From the figure, we can observe that along with the increase of the NTS, the IDG values of all the MOEAs increase as well. Recall that a lower value of IGD means a better quality of solutions. This suggests that when the NTS is increasing, it affects the quality

Table 4. Results of the Spearman's rank correlation test – RQ3

Metric	MOSA				
	IBEA	NSGA-II	NSGA-III	MOCell	SPEA2
ρ	0.936	0.933	0.930	0.713	0.930
p-value	<0.05	<0.05	<0.05	<0.05	<0.05

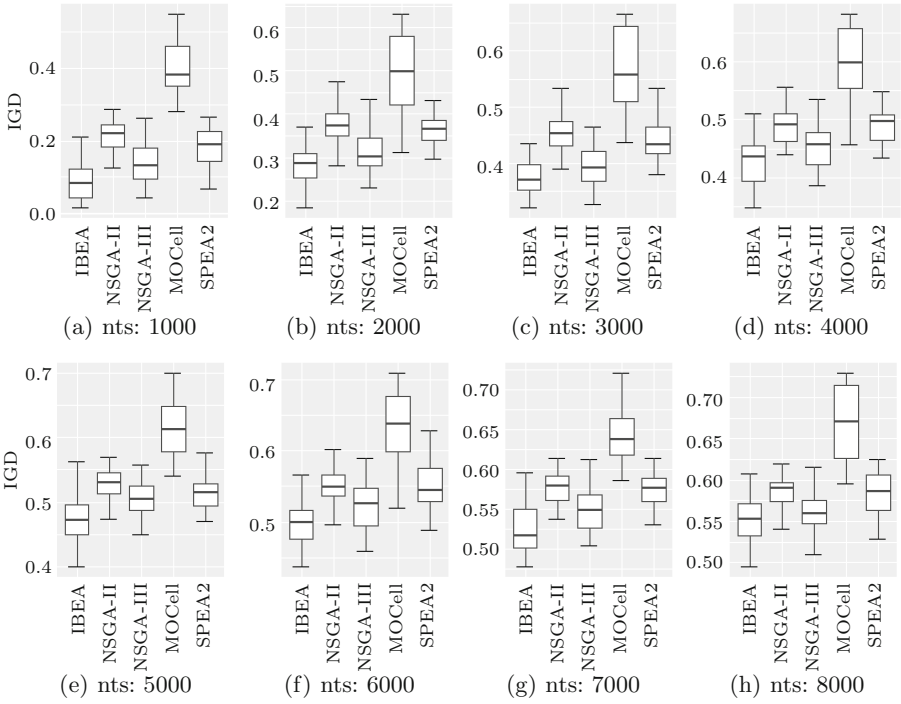

Fig. 3. Descriptive statistics of IGD in terms of various NTS – RQ2

of solutions produced by each MOEA. This is because an optimization problem with a larger NTS has a larger search space (Sect. 2); therefore, the problem is more challenging to solve.

We further studied the statistical significance of this pattern with the Spearman's rank correlation (ρ) test. Results are shown in Table 4, from which we can see that all the ρ values are near 1.0, except for MOCell. This tells that there is a near perfect positive correlation between IGD and NTS. For MOCell, the ρ value is 0.713, indicating a strong positive correlation. In addition, all p-values are less than 0.05, suggesting that this positive correlation is statistically significant.

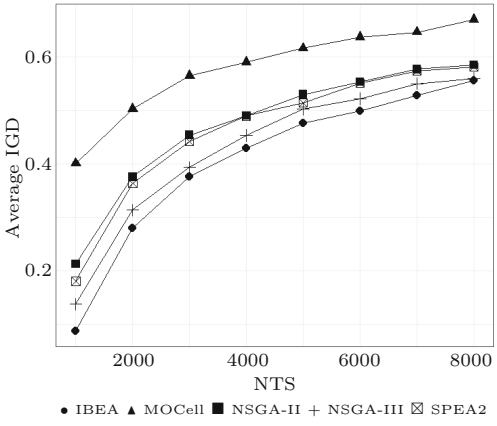

Fig. 4. Results of IGD of various MOEAs when increasing NST – RQ3

These results suggest that the ability of the MOEAs producing high-quality solutions significantly decreases with the increase of NTS. One reason could be that with while increasing NTS, the optimization problem is getting more complex, and hence MOEAs need more generations to find good quality solutions. This aspect needs to be assessed with additional experiments in the future.

4.4 Results of RQ4

Table 5 shows that a MOEA needs nearly 17 to 137 min to solve the optimization problems of different complexity (i.e., NTS). When looking at each NTS value, we can see that there aren't much time differences among the studied MOEAs that practically matter. For example, the best performed algorithm, i.e., IBEA, for the NTS of 8000, took 126 min, which is only approximately 8 min more than RS (with the best time performance). Practically, such minor time difference doesn't matter. Moreover, since SPECTRE is executed offline, it is not critical to select a MOEA in terms of its time performance and we care more about the quality of produced solutions. In summary, the time performance of SPECTRE is acceptable in practice, as, e.g., IBEA spending 17.6 min on selecting and prioritizing 1,000 out of 60,000 test scenarios.

Table 5. Average running time of each MOEA (Time Unit: Minute) – RQ4

MOSA	NTS							
	1000	2000	3000	4000	5000	6000	7000	8000
IBEA	17.58	32.09	47.05	62.42	77.03	98.61	110.15	126.01
NSGA-II	16.19	30.89	46.14	61.87	76.82	97.35	108.92	122.09
NSGA-III	16.21	31.04	46.34	61.41	77.12	97.39	109.19	125.16
MOCell	17.36	33.01	49.77	66.52	82.22	106.89	118.56	136.68
SPEA2	16.81	31.10	46.23	61.81	77.73	97.93	111.35	126.14
Random	14.82	28.56	42.98	56.27	72.14	91.51	102.79	117.68

4.5 Overall Discussion

Research Implications. Based on the evaluation results, we recommend IBEA for SPECTRE as IBEA achieves the best performance among all the selected MOEAs (RQ1 and RQ2). When looking into the performance of the selected MOEAs along with the increase in NTS, IGD values of all the MOEAs increase. This tells that when increasing NTS, the search space increases as well, hence possibly requiring more generations or evaluations to explore good solutions. Moreover, we observed that running time (RQ4) increases with the increase of NTS because of the increased search space and the cost of calculating all the five objectives. For example, as shown in Table 5, for IBEA, selecting 8000 scenarios (126.01 mins) takes about 7 times the running time of selecting 1000 scenarios (17.58 mins). Thus, improving SPECTRE's scalability is one of our future works.

SPECTRE can also be extended to address other optimization problems in testing ADSs, for instance, considering the uncertainty aspect of the attributes of test scenarios. The current implementation of SPECTRE has four attributes, among which CPT (Sect. 2.1) is naturally an uncertain factor. In the future, we can develop solutions by taking into account the uncertain nature of these

attributes. Moreover, SPECTRE can be easily extended by introducing new optimization objectives of newly introduced attributes.

Practical Implications. In practice, testing a new version of an ADS with all existing test scenarios is costly due to limited time-wise and monetary-wise resources. Hence, it is important to have an approach like SPECTRE for cost-effectively construct a new test suite by benefiting from prior knowledge on the cost-effectiveness of already executed test scenarios. SPECTRE requires an organization to construct and maintain a test scenario repository for test optimization (like the one described in Sect. 2.1). This requirement is reasonable as a lot of organizations always have their test management tools (e.g., Bugzilla).

SPECTRE selects and prioritizes a subset of test scenarios out of all available ones based on their attributes (Sect. 2.1) to avoid arbitrary decisions of intuition and experience. Without SPECTRE, the selection and prioritization process may mainly rely on managers' experience and domain expertise.

4.6 Threats to Validity

We employed quality indicator *IGD* to assess the performance of the MOEAs, which is comparable among the selected MOEAs. The total number of evaluations (i.e., 30,000) was used as the stopping criterion for all the MOEAs. We used the same cost measure, i.e., NTS, and repeated the experiments with a given NTS. Regarding parameter settings of the MOEAs, we however mostly used the default parameter settings of the MOEAs from [9], which have shown good results for various SBSE problems [3].

We followed a rigorous statistical procedure to analyze the collected data. Thirty independent runs of each MOEA were performed based on existing guidelines to deal with MOEA's randomness [2]. We chose appropriate tests, i.e., the Mann-Whitney U test with the Bonferroni Correction and $\hat{A}_{12}$ to assess effect based on the established guideline from [2]. We choose an appropriate quality indicator (i.e., IGD) based on the guideline from [1]. Generalization of the results is a key issue with all experiments. Our experiments were conducted with one dataset of 60,000 test scenarios, which is sufficiently large. Nonetheless, additional datasets are needed for future investigation.

5 Related Work

Search-Based Testing (SBT) of ADSs. Ben Abdessalem et al. [4] used NSGA-II and surrogate models to identify most critical behaviors of Advanced Driver Assistance Systems (ADASs) used in ADSs within limited computation resources. They experimented with an industrial ADAS and results show that their approach can automatically identify test cases indicating critical ADAS behaviors with higher quality compared to random search, especially under resource sensitive situations. Another related work combines multi-objective search and decision tree classification models, i.e., NSGAII-DT [5]. This approach tests vision-based control systems of ADSs to find critical scenarios quickly and

accurately. NSGAII-DT was compared with NSGA-II on an industrial ADS, and the results shows that NSGAII-DT significantly outperforms NSGA-II in terms of generating more distinct and critical test scenarios.

FITEST [6] is an SBT approach for detecting feature interaction failures in autonomous driving. FITEST has a combination of traditional coverage based heuristics and novel heuristics specifically proposed for revealing feature interaction failures. They evaluated FITEST using two versions of an industrial ADS. Results show that FITEST is effective in terms of identifying more feature interaction failures than approaches with coverage-based and failure-based test objectives. Gambi et al. [10] addressed challenges of simulation-based testing by combining procedural road generation and genetic algorithms to generate virtual roads, by utilizing *procedural content generation*, for testing ADSs. Their evaluation on two different ADSs shows that the proposed method can generate effective virtual road networks, which can lead car departures from lanes. Li et al. [12] proposed AV-FUZZER which combines genetic algorithms with a local fuzzer to increase the ability of generating AV safety violation scenarios. The approach was evaluated on Baidu Apollo and was proven to be able to find safety violations in a short time.

All the above-mentioned related research focus on generating test scenarios for an ADS, but none of them study the reuse of test scenarios obtained from previous versions of the ADS and used for testing its new version. We, instead, focus on reusing existing test scenarios by selecting and prioritizing them from a repository of test scenarios that have been executed for testing old versions of an ADS, with multi-objective search.

Search-Based Test Case Prioritization (TCP). TCP approaches consider various prioritization criteria such as fault detection rate, time cost estimates, and code coverage. Search-based techniques have been widely used in addressing TCP problems. For instance, in [13], Li et al. described five algorithms for the sequencing problem in test case prioritization for regression testing, defined a fitness function with three objectives, and prioritized test cases with hill climbing and GAs. To study TCP problems with limited budget, many works have been proposed. For instance, Singh et al. [17] used ant colony optimization to prioritize test cases under limited time and cost constrains, with the goal to maximum the number of faults to cover and minimum time cost. Wang et al. [18] proposed a multi-objective search-based prioritization approach, aiming at prioritizing test cases with limited test resources and time budget, with one cost measure on test execution time and three effectiveness measures on prioritization density, test resources usage and fault detection capability. TCP approaches focusing on the fault detection capability also attracted much attention. For instance, Pradhan et al. [15] proposed a multi-objective approach to prioritize test cases with the high coverage of configurations, test APIs, statuses, and high fault detection capability as quickly as possible. Moreover, Luo et al. [14] conducted an empirical study to evaluate the performance of a TCP approach in terms of its fault detection efficiency.

Though, our aim is similar to these works, i.e., test prioritization for regression testing, we focus on prioritization on test scenario prioritization for ADSs, which is not much studied in the literature.

6 Conclusion and Future Work

This paper presented a multi-objective search approach for test scenario selection and prioritization for autonomous driving systems (ADSs) with the ultimate aim of decreasing the testing cost of newer versions of ADSs based on historical test data collected for testing their previous versions. The approach integrated five multi-objective evolutionary algorithms (MOEAs), i.e., NSGA-II, NSGA-III, IBEA, SPEA2, and MOCell. To evaluate our approach with the selected MOEAs, we experimented with one large-scale dataset collected from testing an open-source ADS. The dataset is of 60,000 test scenarios. Our evaluation results showed that IBEA performed the best in terms of producing high quality solutions, and therefore is recommended for addressing our optimization problem.

In the future, we will test other ADSs to assess SPECTRE's performance and scalability. Moreover, attribute values associated with each scenario are required to be continuously updated due to continuous testing of newer versions of ADSs. Thus, we will provide a framework for automatically updating these values together with an online repository. We will also perform parameter tuning for MOEAs and study their effect on the performance of our approach.

Acknowledgements. The work is supported by the National Natural Science Foundation of China under Grant No. 61872182. The work is also partially supported by the Co-evolver project (No. 286898/F20) funded by the Research Council of Norway. Huihui Zhang is supported by the Science and Technology Program of Public Wellbeing (No. 2020KJHM01).

References

1. Ali, S., Arcaini, P., Pradhan, D., Safdar, S.A., Yue, T.: Quality indicators in search-based software engineering: an empirical evaluation. ACM Trans. Softw. Eng. Methodol. (TOSEM) **29**(2), 1–29 (2020)
2. Arcuri, A., Briand, L.: A practical guide for using statistical tests to assess randomized algorithms in software engineering. In: Proceedings of the 33rd International Conference on Software Engineering (ICSE 2011), pp. 1–10 (2011)
3. Arcuri, A., Fraser, G.: On parameter tuning in search based software engineering. In: Cohen, M.B., Ó Cinnéide, M. (eds.) SSBSE 2011. LNCS, vol. 6956, pp. 33–47. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23716-4_6
4. Ben Abdessalem, R., Nejati, S., Briand, L.C., Stifter, T.: Testing advanced driver assistance systems using multi-objective search and neural networks. In: Proceedings of the Conference on Automated Software Engineering, pp. 63–74. ACM (2016)
5. Ben Abdessalem, R., Nejati, S., Briand, L.C., Stifter, T.: Testing vision-based control systems using learnable evolutionary algorithms. In: Proceedings of the Conference on Software Engineering, pp. 1016–1026. ACM (2018)

6. Ben Abdessalem, R., Panichella, A., Nejati, S., Briand, L.C., Stifter, T.: Testing autonomous cars for feature interaction failures using many-objective search. In: Proceedings of the Conference on Automated Software Engineering, pp. 143–154. ACM (2018)
7. Corso, A., Du, P., Driggs-Campbell, K., Kochenderfer, M.J.: Adaptive stress testing with reward augmentation for autonomous vehicle validatio. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp. 163–168. IEEE (2019)
8. Czarnecki, K.: Operational design domain for automated driving systems: Taxonomy of basic terms. Waterloo Intelligent Systems Engineering (WISE) Lab, University of Waterloo, Canada (2018)
9. Durillo, J.J., Nebro, A.J.: jMetal: a Java framework for multi-objective optimization. Adv. Eng. Softw. **42**(10), 760–771 (2011)
10. Gambi, A., Mueller, M., Fraser, G.: Automatically testing self-driving cars with search-based procedural content generation. In: Proceedings of International Symposium on Software Testing and Analysis, pp. 318–328. ACM (2019)
11. Greer, D., Ruhe, G.: Software release planning: an evolutionary and iterative approach. Inf. Softw. Technol. **46**(4), 243–253 (2004)
12. Li, G., et al.: AV-FUZZER: finding safety violations in autonomous driving systems. In: International Symposium on Software Reliability Engineering, pp. 25–36. IEEE (2020)
13. Li, Z., Harman, M., Hierons, R.M.: Search algorithms for regression test case prioritization. IEEE Trans. Softw. Eng. **33**(4), 225–237 (2007)
14. Luo, Q., Moran, K., Poshyvanyk, D., Di Penta, M.: Assessing test case prioritization on real faults and mutants. In: 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 240–251. IEEE (2018)
15. Pradhan, D., Wang, S., Ali, S., Yue, T., Liaaen, M.: STIPI: using search to prioritize test cases based on multi-objectives derived from industrial practice. In: Wotawa, F., Nica, M., Kushik, N. (eds.) ICTSS 2016. LNCS, vol. 9976, pp. 172–190. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47443-4_11
16. Ramirez, A., Romero, J.R., Ventura, S.: A survey of many-objective optimisation in search based software engineering. Syst. Softw. Eng. **149**, 382–395 (2019)
17. Singh, Y., Kaur, A., Suri, B.: Test case prioritization using ant colony optimization. ACM SIGSOFT Softw. Eng. Notes **35**(4), 1–7 (2010)
18. Wang, S., Ali, S., Yue, T., Bakkeli, Ø., Liaaen, M.: Enhancing test case prioritization in an industrial setting with resource awareness and multi-objective search. In: Proceedings of the 38th International Conference on Software Engineering Companion, pp. 182–191 (2016)
19. Yoo, S., Harman, M.: Pareto efficient multi-objective test case selection. In: Proceedings of the International Symposium on Software Testing and Analysis, pp. 140–150 (2007)
20. Zhang, H., Zhang, M., Yue, T., Ali, S., Li, Y.: Uncertainty-wise requirements prioritization with search. ACM Trans. Softw. Eng. Methodol. (TOSEM) **30**(1), 1–54 (2020)
21. Zhang, M., Ali, S., Yue, T.: Uncertainty-wise test case generation and minimization for cyber-physical systems. J. Syst. Softw. **153**, 1–21 (2019)
22. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Yao, X., et al. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 832–842. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30217-9_84