Stability Analysis for Safety of Automotive Multi-Product Lines: A Search-Based Approach

Nian-Ze Lee
National Taiwan University
Taipei, Taiwan
d04943019@ntu.edu.tw

Paolo Arcaini
National Institute of Informatics
Tokyo, Japan
arcaini@nii.ac.jp

Shaukat Ali
Simula Research Laboratory
Oslo, Norway
shaukat@simula.no

Fuyuki Ishikawa
National Institute of Informatics
Tokyo, Japan
f-ishikawa@nii.ac.jp

ABSTRACT

Safety assurance for automotive products is crucial and challenging. It becomes even more difficult when the variability in automotive products is considered. Recently, the notion of *automotive multi-product lines* (multi-PL) is proposed as a unified framework to accommodate different sources of variability in automotive products. In the context of automotive multi-PL, we propose a *stability* analysis for safety, motivated by our industrial collaboration, where we observed that under certain operation scenarios, safety varies drastically with small fluctuations in production parameters, environmental conditions, or driving inputs. To characterize instability, we formulate a multi-objective optimization problem, and solve it with a search-based approach. The proposed technique is applied to an industrial automotive multi-PL, and experimental results show its effectiveness to spot instability. Moreover, based on information gathered during the search, we provide some insights on both testing and quality engineering of automotive products.

CCS CONCEPTS

• **Computer systems organization → Embedded and cyber-physical systems**; • **Software and its engineering → Search-based software engineering**;

KEYWORDS

Safety, Robustness, Stability Analysis, Automotive Domain, Product Line, Simulink, Search

ACM Reference Format:
Nian-Ze Lee, Paolo Arcaini, Shaukat Ali, and Fuyuki Ishikawa. 2019. Stability Analysis for Safety of Automotive Multi-Product Lines: A Search-Based Approach. In *Genetic and Evolutionary Computation Conference (GECCO '19), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3321707.3321755

GECCO '19, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6111-8/19/07…$15.00
https://doi.org/10.1145/3321707.3321755

1 INTRODUCTION

Safety is the most essential concern in the automotive industry and must be ensured. The guarantee of safety becomes even more challenging when different types of the variability of automotive products are considered. The notion of a *product line* (PL) is a way to capture variability within a family of products that share some common characteristics, but differ in others [21]. PL engineering includes the development, maintenance, and analysis of PLs by taking into account their commonalities and variability. To tackle the complexity of heterogeneous systems in which multiple PLs interact, the notion of a *multi-product line* (multi-PL) has been proposed [15]. In this paper, we apply the concept of multi-PLs to the automotive domain (referred to as *automotive multi-PL*) to account for the different kinds of variability of the automotive products. This can help to analyze how variability affects the behavior of automotive products. In an automotive multi-PL, variability is captured by *parameters* which describe the production specification of the product, the operation scenarios of the product, and the inputs given to the product. A *configuration* of an automotive multi-PL is an assignment of concrete values to all the parameters which are used to depict the variability.

Prior research endeavors of safety assurance for automotive products mostly focus on *falsification* [16], which aims at identifying an operation scenario where an automotive product violates a safety requirement. In our work, motivated by industrial collaboration experience, we propose a *stability analysis* for safety, in the context of automotive multi-PLs. In an industrial case study, we observed that the safety of an automotive product is greatly affected by small changes in its configuration. Specifically, in the case study, we investigated the effect of an active safety feature on the safety of a car. The active safety feature is triggered when an emergency braking happens, and will decide whether and when to shut down the engine if it is necessary to avoid an accident. We found a configuration where an accident is unavoidable if the active safety feature decides to shut down the engine 0.08 seconds late. The observation that only 2.8% variation in a single parameter could make the difference between a safe stop and a crash surprises our industrial partner. We emphasize that the information of the instability is helpful for the engineers to design more robust automotive products, as it reflects the influence of parameters on safety. However, spotting instability in an automotive multi-PL is difficult,

as it involves searching in the huge configuration space. Moreover, clearly specifying the desired instability to spot requires domain knowledge, and in general is not an easy task.

To automatically and efficiently discover the instability of safety without assuming domain knowledge, we propose an approach which captures the instability by formulating a multi-objective optimization problem, where the difference between two configurations is minimized and their safety difference is maximized. It solves the multi-objective optimization problem by a search-based approach, which computes a *Pareto front* of the problem. The Pareto front shows how the difference in safety grows as the configuration difference grows, giving an overview of how they interact with each other. The approach has been implemented, using NSGA-II [9] as the underlying search algorithm, and applied to an industrial automotive multi-PL Simulink[1] model. The proposed technique effectively computes the Pareto front, providing useful information to pinpoint the instability of the Simulink model. Moreover, we record the simulation data during the search for further analyses. The results can potentially provide insights on testing and quality engineering of automotive products. Our contributions include:

- An approach of safety analysis for automotive multi-PLs, which relies on a multi-objective optimization problem to capture the instability in automotive multi-PLs, and does not assume domain knowledge. It is able to provide an overview about how safety is affected by configurations.
- Applying the proposed technique to an industrial Simulink model, where the technique significantly outperforms random search and effectively spots the instability of the model.

The rest of the paper is organized as follows. After giving background knowledge in Section 2, we introduce our problem formulation in Section 3. The industrial case study is detailed in Section 4, and a search-based approach to analyze the stability of safety is presented in Section 5. Section 6 demonstrates experimental results to show the effectiveness of the approach, and Section 7 discusses possible threats to its validity. Finally, we review related literature in Section 8, and conclude the paper in Section 9.

2 BACKGROUND

In this section, we introduce necessary background knowledge to facilitate subsequent discussion.

2.1 Modeling Hybrid Systems

A *hybrid system* takes real-valued input signals and generates real-valued output signals. Suppose a hybrid system has n input ports and m output ports. An input (resp. output) signal u (resp. v) for the hybrid system is a mapping $u : [0, \infty) \to \mathbb{R}^n$ (resp. $v : [0, \infty) \to \mathbb{R}^m$). A *model* M of the hybrid system is a function that maps an input signal u to an output signal v.

Hybrid systems appear ubiquitously and in principle can be modeled with different programming languages. An example of common hybrid systems is automotive products, e.g., cars or their components. In the automotive industry practice, Simulink is widely adopted to model automotive products, and is also the primary formalism of our work.

[1] Simulink™ is a registered trademark of the MathWorks Inc.

2.2 Quantitative Safety and Falsification

Safety is the most crucial concern for automotive products. As an automotive product has real-valued input/output signals, its safety is also measured in a *quantitative* way. For example, in a crash test of cars, the crashing speed is often used as a quantitative measure of how severely safety is violated. We call the quantitative safety measurement *robustness*, and will give an example in Section 4.3.

The concept of robustness can be made precise by adopting the *robustness semantics* of *signal temporal logic* (STL) [10], which extends *linear temporal logic* with real-time and real-valued signals. Using STL to specify a safety requirement, one can derive a function to compute the robustness value. Given a model M and an STL formula φ, the robustness computation function derived from the robustness semantics is denoted as $rob_{M,\varphi}(\cdot)$, which takes an input signal u and returns the robustness value of u with respect to φ. The robustness value $rob_{M,\varphi}(u)$ describes how *strongly* φ is satisfied by u when M receives u as input. Intuitively speaking, when $rob_{M,\varphi}(u)$ is positive, it means φ is satisfied by u. On the other hand, negative $rob_{M,\varphi}(u)$ indicates that φ is violated. The absolute value of $rob_{M,\varphi}(u)$ reflects the *degree* of satisfaction or violation. That is, a greater magnitude of the robustness value means that the property is strongly satisfied (resp. severely violated) if the robustness value is positive (resp. negative). We refer readers to [10] for a rigorous discussion of the robustness semantics of STL.

Based on the notion of robustness, *falsification* is a state-of-the-art validation approach for hybrid systems. Given a model M and an STL formula φ, it tries to find an input signal u such that $rob_{M,\varphi}(u) < 0$. It has been intensively studied in recent years [16].

3 PROBLEM FORMULATION

In this section, we first introduce the variability of automotive products, and lift the concept of modeling an automotive product discussed in Section 2.1 to modeling a whole automotive multi-PL. Second, we formulate a multi-objective optimization problem as a way to capture the notion of instability in an automotive multi-PL.

3.1 Variability of Automotive Multi-PLs

Our work started from the observation that, in the automotive domain, there exist different kinds of variability which influence the behavior of automotive products. Three types of variability in automotive products are identified. The first type is related to the *production parameters* of the car. For example, different models in the same car family can have different versions of some component, e.g., engines with different horsepower and torque. Also, some component, such as an active safety feature, may exist only on a high-end car model. The second type of variability comes from the *environmental conditions*. As an example, the type of roads or the atmospheric conditions (e.g., rain, wind, etc.) could affect a car's braking functionality. Different driving scenarios, for example, on a highway or in a city, also fall into this type of variability. The third type of variability is associated with the *driving behavior*, i.e., the inputs given to the car, such as acceleration, braking, or steering. Characteristics of the driver, e.g., the reaction time to push the brake pedal when an accident happens in front of the car, also belong to this category.

We propose to view the car model as a *multi-product line* (multi-PL) [4, 15] where variability occurs in the production specification of the car, environmental conditions, and driving behavior. We claim that explicitly modeling and considering these kinds of variability is beneficial, as it allows to understand how the behavior of automotive products changes with respect to variability.

Let M be the model of the automotive multi-PL under analysis, which has *parameters* $P = (P_1, \ldots, P_n)$. Based on our previous observations, we distinguish three types of parameters:

- *production parameters*: related to the production specification of the automotive product;
- *environmental parameters*: related to the environment in which the automotive product is operated;
- *behavioral parameters*: inputs to the automotive product during driving given either by a human driver or by the controller of an autonomous car.

Among the parameters P, there might be constraints imposed by the automotive multi-PL to exclude infeasible configurations. For example, a constraint could restrict the relation between the weight of the car and the horsepower of the engine by requiring that, if the weight of the car is greater than 1300 kg, the horsepower must be greater than 120 kW.

In the following discussion, we treat parameters as constant input signals of a model M, and denote the robustness of a configuration c with respect to an STL formula φ as $rob_{M,\varphi}(c)$, which is further simplified to $rob(c)$ for readability.

3.2 Instability of Automotive Multi-PLs

We define the notion of *instability* in a model from two essential aspects: robustness difference and configuration difference. If a pair of *similar* configurations exhibits quite *different* behaviors in terms of safety (e.g., it has a large difference in robustness), this pair of configurations is a witness of the instability. In practice, defining such "similarities" and "differences" requires domain knowledge. A similar circumstance about the assumption of domain knowledge also appears in [13], where the *next release problem* is studied under two different settings: the first one assumes requirement engineers have knowledge about the expected inaccuracy of budget estimations, while the second one does not make this assumption.

Instead of relying on domain experts to specify a criterion of instability, we formulate a multi-objective optimization problem, where robustness difference is maximized and configuration difference is minimized, to find instability in an automotive multi-PL. We remark that the authors in [13] also resort to multi-objective optimization to solve their second setting which assumes no domain knowledge from the users.

Problem Statement 1 (Capturing instability by multi-objective optimization). *Given a model M of an automotive multi-PL, let Σ be the set of feasible configurations specified by constraints of M, c, d be two configurations, and $D(c, d)$ be the configuration difference between c and d. The multi-objective optimization problem to capture instability in M is defined as:*

$$\max_{c,d \in \Sigma} |rob(c) - rob(d)|$$
$$\min_{c,d \in \Sigma} D(c, d)$$

In the above formulation, if all parameters in M are real numbers, one can use Euclidean distance for the distance function $D(\cdot, \cdot)$.

We remark that using the proposed multi-objective optimization formulation to capture instability has two benefits. First, it does not rely on users to define instability. Second, the computed Pareto front reflects how the maximum found robustness difference responds to the increase in the configuration difference, providing users with an overview of how they interact with each other.

3.2.1 Classification based on safety. Observe that in Problem Statement 1, only the difference between $rob(c)$ and $rob(d)$ is considered, but their signs are overlooked. However, the signs of $rob(c)$ and $rob(d)$ are important, as they indicate whether the safety requirement is satisfied or violated. Therefore, we classify a pair of configurations (c, d) based on the signs of $rob(c)$ and $rob(d)$ into the following three categories:

- safe-unsafe: $rob(c) \times rob(d) < 0$. We claim that this is the most critical case among the three categories, as it shows how changing some parameter values makes the difference between satisfaction and violation of the safety requirement. We name a pair of configurations belonging to this category an *unstable pair of configurations*, and a configuration in this pair an *unstable configuration*. In the following discussion, we mainly focus on unstable pairs of configurations.
- unsafe-unsafe: $rob(c) \leq 0$ and $rob(d) \leq 0$. In this case, although the safety requirement is violated in both configurations, the *severity* is lower in one of the two. Therefore, pairs of configurations belonging to this category might shed light on risk management when an accident is unavoidable.
- safe-safe: $rob(c) > 0$ and $rob(d) > 0$. While in this case both configurations satisfy the safety requirement, they might have potential to help engineers understand how safety is affected by variability of the automotive multi-PL.

3.2.2 Limiting the form of configuration difference. From our industrial collaboration experience, we observed that in many unstable pairs of configurations, the two configurations often differ in only one parameter value. These pairs of configurations show the contribution of a single car component, environmental phenomenon, or behavioral input to the safety. On the other hand, if multiple parameters change in the unstable pairs of configurations, it becomes difficult to assess the contribution of a single parameter. As a result, currently we focus on investigating pairs of configurations which only differ in one parameter, and leave the case where multiple parameters vary together as a future work.

In the next section, we will describe a model of an industrial automotive multi-PL, and a search-based approach to solve Problem Statement 1 will be presented in Section 5.

4 INDUSTRIAL CASE STUDY

In this section, we concretize the ideas discussed in Section 3 by showing a Simulink model, which describes an automotive multi-PL. The Simulink model is built by our industrial partner to design an active safety feature in the cars. We will introduce the parameters, outputs, constraints, safety requirement, and robustness evaluation of the model.

Table 1: Model parameters, types of variability, and domains

Parameter	Type	Unit	Domain
T_safe	production	second	[0, 5]
Radius_tire	production	meter	[0.3, 0.37]
Power_max	production	kW	[70, 130]
Torque_max	production	Nm	[150, 450]
Weight_car	production	kg	[800, 1500]
T_init	environmental	second	[1, 5]
A_back	behavioral	G	[0.1, 1]
T_behav	behavioral	second	[0.5, 5]
V_init	behavioral	km/h	[20, 160]
R_gear	behavioral	-	{2.5, 4, 5.5, 7, 8.5, 10}

4.1 Model Description

The model describes the following simulation scenario: a car drives on a straight lane and spots an obstacle in front of it. When the driver pushes the brake pedal, an active safety feature is also triggered and will decide whether and when to shut down the engine if it is necessary to avoid an accident. The parameters of the model as well as their types of variability and domains are listed in Table 1.

The initial distance between the car and the obstacle is the product of the initial inter-vehicle time `T_init` (second) and initial car speed `V_init` (km/h). Upon spotting the obstacle, the driver will push the brake pedal after `T_behav` (second), and the active safety feature will shut down the engine after `T_safe` (second). Notice that, although `T_safe` is decided by the active safety feature, the introduced Simulink model takes it as a free input. The evasive backward acceleration is `A_back` (G) and the ratio between engine RPM (Revolutions Per Minute) and wheel RPM is `R_gear`. The car has tires with a radius equal to `Radius_tire` (meter), an engine with the maximum horsepower and torque equal to `Power_max` (kW) and `Torque_max` (Nm), respectively, and weights `Weight_car` (kg). Note that `R_gear` is discrete, while the others are continuous.

The model is composed of 4 subsystems handling different functionalities: (1) *propulsion* subsystem computes the longitudinal acceleration of the vehicle; (2) *inter-obstacle* subsystem computes the distance and collision speed with the obstacle; (3) *steering* subsystem computes the lateral acceleration of the vehicle; (4) *deviation* subsystem computes the deviation speed. In total, the Simulink model consists of 22 blocks and a Matlab function of 65 lines. Using the code generation functionality of Simulink, the model is converted to roughly 1000 lines of C code.

Given a configuration of the model, we simulate the model and observe the outcome. There are two possibilities: either the car stops in front of the obstacle, or it crashes into the obstacle. These two outcomes are captured by two model outputs, `distance` (meter) and `V_collision` (km/h). Output `distance` indicates the final distance between the car and the obstacle, and output `V_collision` indicates the speed with which the car crashes into the obstacle if a collision happens. These two outputs will be used to define the robustness of a configuration of the model in the following discussion.

4.2 Constraints

As the model describes an automotive multi-PL, there exist constraints over production parameters. Specifically, these constraints are over `Radius_tire`, `Power_max`, `Torque_max`, and `Weight_car`. The spirit behind these constraints is to reflect the design rationale that heavy cars often have larger tires and stronger engines, and vice versa. These constraints, listed below, should be taken into account in any analysis of the model.

$$\texttt{Weight\_car} > 1300 \implies \texttt{Radius\_tire} > 0.34$$
$$\texttt{Weight\_car} > 1300 \implies \texttt{Power\_max} > 120$$
$$\texttt{Weight\_car} > 1300 \implies \texttt{Torque\_max} > 200$$
$$\texttt{Weight\_car} < 1200 \implies \texttt{Radius\_tire} < 0.33$$
$$\texttt{Weight\_car} < 1200 \implies \texttt{Power\_max} < 100$$
$$\texttt{Weight\_car} < 1200 \implies \texttt{Torque\_max} < 300$$

4.3 Safety Requirement and Robustness

The safety requirement for this model, stated in the natural language, is as follows. *"The car should not crash into the obstacle; if a collision is unavoidable, the car should collide at the lowest possible speed."* Based on this requirement, below we devise a robustness computation function to evaluate safety quantitatively.

Denote the model by M and a configuration of M by c. The output values $\texttt{distance}(c)$ and $\texttt{V\_collision}(c)$ are obtained by simulating M with c. The robustness computation function $rob(\cdot)$ is defined as follows.

$$rob(c) = \begin{cases} \texttt{distance}(c), & \text{if } \texttt{distance(c)} > 0, \\ \texttt{-V\_collision}(c), & \text{otherwise.} \end{cases}$$

5 STABILITY ANALYSIS FOR SAFETY

In this section, we introduce the proposed stability analysis for safety. It analyzes the stability for safety of a model through solving the multi-objective optimization problem formulated in Problem Statement 1. Search algorithms are employed as underlying solving techniques to tackle multi-objective optimization, as they are shown to be efficient and effective in various multi-objective optimization problems [14].

As discussed in Section 3.2.2, we focus on the case where the two configurations only differ in one parameter P_i. We call P_i the *analyzed parameter*, which can be specified by users. The proposed technique takes a model M and its parameters $(P_1, \ldots, P_n)$, a function $rob(\cdot)$ which maps a configuration to its robustness value, and an analyzed parameter P_i as input. It returns a Pareto front computed by the search as output.

In the following, we will describe the setup of the search, including the encoding of an individual, the handling of constraints, and the design of fitness functions.

5.1 Encoding of An Individual

An individual in the search represents a decision vector in the multi-objective optimization problem. In Problem Statement 1, a decision vector is a pair of configurations $c = (v_1, \ldots, v_n)$ and $d = (v'_1, \ldots, v'_n)$, where for $j \in \{1, \ldots, n\}$, value v_j (resp. value v'_j)

is the value assigned to parameter P_j in configuration c (resp. configuration d). As we only allow c and d to differ in the values of parameter P_i, i.e., v_i and v_i', an individual x to represent a pair of configurations c and d is encoded as $(v_i, v_i', v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$. Following the classification of pairs of configurations discussed in Section 3.2.1, an individual is said to be an *unstable individual* if it consists of an unstable pair of configurations.

5.2 Handling of Constraints

As explained in Section 3.1, an automotive multi-PL might have constraints over its parameters to exclude infeasible configurations. Therefore, we have to handle these constraints during the search. While different approaches have been proposed for handling constraints during the search [12], we follow a *static penalty* approach [12] to compute the *degree of violation* of a configuration with respect to a constraint. In the approach, the degree of violation is normalized between $[0, 1]$. If the degree of violation equals 0, it means the configuration satisfies the constraint. A greater degree of violation means a worse violation of the constraint. If there exist multiple constraints, we use the average of the degree of violation with respect to each constraint to represent the degree of violation a configuration. Recall that in our search, an individual x consists of a pair of configurations c and d. Therefore, the degree of violation for x is computed as the average of the degree of violation for c and d. In the following, we use $dv(x)$ to denote the degree of violation of an individual x. Note that $dv(x)$ also ranges between 0 and 1.

5.3 Design of Fitness Functions

Given an individual x consisting of two configurations c and d, we evaluate its fitness according to the two objectives in Problem Statement 1, i.e., maximizing robustness difference and minimizing configuration difference, by designing two corresponding fitness functions. To handle the constraints, we involve the degree of violation into fitness computation to guide the search towards finding feasible configurations.

5.3.1 Robustness difference. For simplicity, we denote the robustness values of configurations c and d, $rob(c)$ and $rob(d)$, by r_c and r_d, respectively. Fitness f_r of individual x is computed as

$$f_r(x) = \begin{cases} 0.5 \times (1 + dv(x)) & \text{if } dv(x) > 0, \\ \frac{0.5}{1 + w(r_c, r_d) + |r_c - r_d|} & \text{otherwise.} \end{cases}$$

Note that in the above fitness function, if $dv(x) > 0$, the fitness value of individual x ranges from 0.5 to 1. In this case, the fitness is computed only with the degree of violation. As the robustness values are not required, we do not simulate the model. On the other hand, if $dv(x) = 0$, we simulate the model under configurations c and d to get their robustness values, and the fitness value is normalized in $[0, 0.5]$. If the fitness value is close to 0, it means a huge robustness difference, and a greater fitness value means a smaller robustness difference.

As discussed in Section 3.2.1, we classify a pair of configurations into three categories. To prioritize these three cases, we introduce a prioritizing term $w(r_c, r_d)$ into the evaluation of the fitness value, which returns a prioritizing weight based on the signs of r_c and r_d. A more favored case will receive a greater weight. In our approach, the order of importance from high to low is *safe-unsafe*, *unsafe-unsafe*, and *safe-safe*. Taking the magnitude of robustness difference of the model described in Section 4 into account, we design $w(r_c, r_d)$ to be:

$$w(r_c, r_d) = \begin{cases} 2000 & \text{if } r_c \times r_d < 0, \\ 1000 & \text{if } r_c \leq 0 \text{ and } r_d \leq 0, \\ 0 & \text{otherwise.} \end{cases}$$

5.3.2 Configuration difference. As we only allow parameter P_i to differ between configurations c and d in individual x, the configuration difference between c and d is evaluated as the normalized difference between parameter values v_i and v_i'. Let the domain of parameter P_i be $[d_i^L, d_i^U]$. Fitness f_c of individual x is computed as

$$f_c(x) = \frac{|v_i - v_i'|}{d_i^U - d_i^L}$$

6 EXPERIMENTAL RESULTS

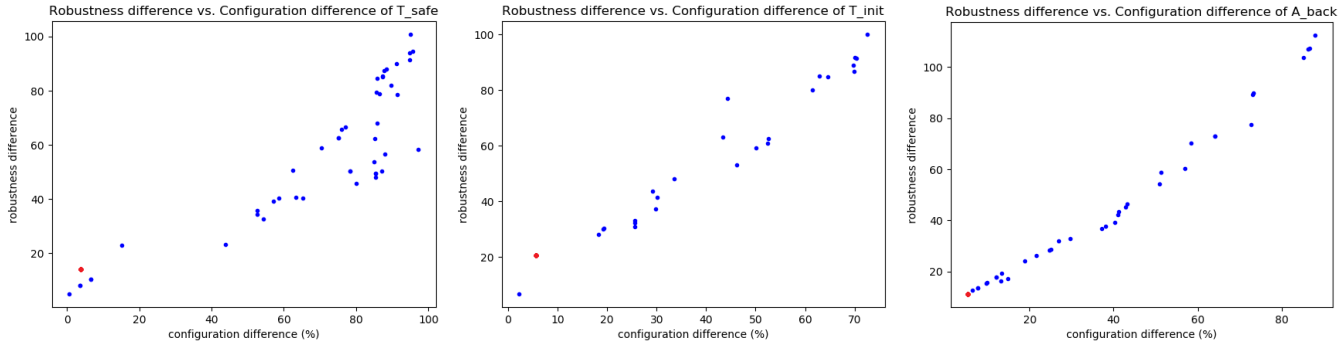
In this section, we present the results of applying the proposed *Stability Analysis for Safety*, abbreviated as SAS, to the Simulink model of an industrial automotive multi-PL described in Section 4. SAS is implemented in the Java™ programming language. The underlying search framework is jMetal 5.6 [11, 17], and algorithm NSGA-II [9] is chosen as the representative of multi-objective search algorithms. We adopt the default algorithmic setting for NSGA-II in jMetal, which includes a population size of 100, parents selection with Binary tournament, a crossover rate of 0.9, and a mutation rate equal to the reciprocal of the number of variables. The default crossover and mutation operations in jMetal are SBX crossover and polynomial mutation, respectively, and the initial population is randomly generated. *Random search* (RS) is chosen as the comparison baseline to justify the use of a complex algorithm. The search will terminate after evaluating the fitness functions for 2000 times. We use SAS to analyze all 10 parameters in the Simulink model.

Note that jMetal provides a native support to handle constraints. However, we choose to handle constraints ourselves, as it allows us to better control the relation between constraint violation and fitness. Our constraints handling proved to be quite effective, as 80% of individuals in the search are feasible.

The experiments are conducted on Amazon Elastic Compute Cloud, using instances with a 2.9 GHz Intel Xeon CPU and 15GB RAM. To account for the randomness in search algorithms, we repeat the experiments for 30 times.

With the above setting of NSGA-II, the average time for SAS to analyze one parameter is about 1000 seconds, and 95% of the time is spent on simulating the model (one simulation execution takes about 0.3 seconds for the model we use) to obtain robustness values. In total, SAS spends about 3 hours analyzing every parameter in the entire automotive multi-PL. The time consumption is considered affordable for our industrial partner, as the analysis is performed over the whole family of products, instead of a single product.

Below, we first compare NSGA-II with RS. Second, we demonstrate the effectiveness of SAS by showing unstable configurations on the Pareto fronts computed by NSGA-II. Third, we record and analyze all the simulation data during the search of NSGA-II, and discuss the potential usage of the analysis results to help model



(a) Production: T_safe **(b) Environment: T_init** **(c) Behavior: A_back**

Figure 1: Visualization of unstable individuals on a Pareto front

testing and quality engineering. Finally, we wrap up this section by discussing some feedback from our industrial partner.

Due to the space limitation, in the following we selectively present the results of production parameter T_safe, environmental parameter T_init, and behavioral parameter A_back as representatives for the three types of variability, respectively. The complete experimental results can be found in the supplementary material.

6.1 NSGA-II Compared with RS in SAS

To justify the usage of NSGA-II in SAS, we compare the results obtained by NSGA-II with those obtained using RS. We choose *hypervolume* (HV) as the evaluation metric to assess the quality of solutions produced by multi-objective search algorithms based on a published guide [22]. To assess the significance of results, we apply the Mann-Whitney U Test and, to assess the strength of the significance, we employ the A12 statistics as the effect size measure. We choose these tests based on the guidelines for selecting appropriate statistical tests for search algorithms reported in [5]. Mann-Whitney U test produces a p-value that determines the significance of the differences between NSGA-II and RS. If a p-value is less than 0.05, then the differences are significant. The A12 statistics produces an A12 value. If the value is equal to 0.5, it means there is no difference between NSGA-II and RS. If the value is greater than 0.5, then NSGA-II is highly likely to be better than RS, and vice versa. For all the 10 experiments (i.e., SAS applied to a given parameter), the Mann-Whitney U test produces p-values less than 0.05 and all the A12 values are greater than 0.5, suggesting that NSGA-II significantly outperforms RS for all the 10 parameters. Detailed statistical data are reported in the supplementary material.

6.2 Effectiveness of SAS

To show that SAS effectively spots instability of the model, we report the unstable individuals on the computed Pareto fronts. In Figure 1, we select an arbitrary Pareto front and plot the robustness difference on the vertical axis versus the configuration difference on the horizontal axis for each unstable individual. Figure 1 shows the maximum found robustness difference under a given configuration difference, outlining the interaction between the robustness difference and configuration difference.

As discussed in Section 3.2, clearly specifying instability in terms of the thresholds for robustness and configuration difference is not an easy task. Instead, SAS computes a Pareto front showing the trade-off between the two competing aspects. From the Pareto front, the designers can pinpoint interesting configurations, and inspect the assigned value for each parameter.

As an example, in each plot of Figure 1, we highlight one individual in red, and report the robustness and parameter values of the individual in Table 2. Configurations $c_1^{\texttt{T\_safe}}$ and $c_2^{\texttt{T\_safe}}$ (resp. $c_1^{\texttt{A\_back}}$ and $c_2^{\texttt{A\_back}}$) show that shutting down the engine 0.123 seconds late (resp. providing a backward acceleration 0.006 G less) could result in an accident. On the other hand, the difference in T_init between configurations $c_1^{\texttt{T\_init}}$ and $c_2^{\texttt{T\_init}}$ equals 0.12 seconds, which results in 2.285 meters (roughly half the length of a car) difference in their initial distances. However, configuration $c_1^{\texttt{T\_init}}$ satisfies the safety requirement, stopping in front of the obstacle by 0.333 meters, while configuration $c_2^{\texttt{T\_init}}$ crashes into the obstacle at a speed of 20.188 km/h. Given the huge search space of automotive multi-PLs, these unstable configurations are not easily detectable. Moreover, as unstable configurations reflect the vulnerability of safety due to small configuration fluctuations, the information can help designers to test or debug the model. Overall, the capability and effectiveness of SAS to spot unstable configurations show its unique value.

6.3 Analyses of Simulation Data

As mentioned above, we record the simulation result for every individual explored during the search. With the obtained data, we wish to provide more information about the model to our industrial collaborator. While various analyses can be adopted, as a starting point we apply quartile analysis, which computes quartiles and the *interquartile range* (IQR) of the given data. The results of quartile analysis could benefit model testing and quality engineering, which will be discussed below.

6.3.1 Benefit for model testing. As we wish to characterize the instability of the model, unstable individuals are selected and classified by their analyzed parameter. This way, each parameter has a list of unstable individuals, where the variation of the parameter itself

Table 2: Robustness and parameter values of highlighted unstable individuals

Config.	*rob*(*c*)	T_safe	Radius_tire	Power_max	Torque_max	Weight_car	T_init	A_back	T_behav	V_init	R_gear
$c_1^{\texttt{T\_safe}}$	0.189	3.132	0.315	128.041	217.062	1235.841	3.063	0.881	4.076	54.775	8.5
$c_2^{\texttt{T\_safe}}$	-13.873	3.255	0.315	128.041	217.062	1235.841	3.063	0.881	4.076	54.775	8.5
$c_1^{\texttt{T\_init}}$	0.333	4.482	0.324	94.084	306.560	1247.602	2.119	0.825	4.263	68.544	10
$c_2^{\texttt{T\_init}}$	-20.188	4.482	0.324	94.084	306.560	1247.602	1.999	0.825	4.263	68.544	10
$c_1^{\texttt{A\_back}}$	0.539	3.366	0.309	120.473	447.615	1208.102	4.547	0.112	1.064	64.748	8.5
$c_2^{\texttt{A\_back}}$	-10.619	3.366	0.309	120.473	447.615	1208.102	4.547	0.106	1.064	64.748	8.5

Table 3: The IQRs of the unstable region for each parameter

Parameter	T_safe	Radius_tire	Power_max	Torque_max	Weight_car	T_init	A_back	T_behav	V_init	R_gear
T_safe	[0.94, 2.36]	[0.32, 0.35]	[90.73, 121.84]	[276.29, 386.00]	[1223.26, 1271.09]	[1.54, 3.72]	[0.24, 0.74]	[3.85, 4.80]	[30.44, 55.37]	[5.5, 10]
T_init	[2.71, 4.46]	[0.31, 0.35]	[89.67, 114.19]	[215.73, 368.48]	[1212.58, 1271.93]	[1.42, 2.41]	[0.32, 0.77]	[2.89, 4.43]	[43.44, 87.38]	[5.5, 10]
A_back	[3.36, 4.69]	[0.31, 0.35]	[85.45, 121.27]	[229.12, 353.15]	[1218.44, 1269.73]	[2.40, 4.39]	[0.14, 0.25]	[1.12, 2.58]	[43.92, 69.46]	[5.5, 10]

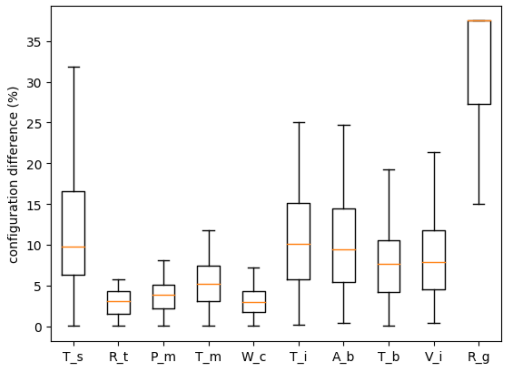
makes the difference between satisfying and violating the safety requirement. To focus on instability caused by small configuration fluctuations, in the list of unstable individuals for each parameter, we analyze the first 1000 individuals with the smallest configuration difference. The configurations in the analyzed unstable individuals capture a region in the configuration space where safety tends to vary drastically when the parameter fluctuates.

We perform quartile analysis on the aforementioned unstable individuals to outline the IQRs of the *unstable region* for each parameter. The results are shown in Table 3, where each row shows the IQRs of the unstable region for the parameter specified in the first column. These results have the potential to help the testing of the model. For example, if a testing engineer would like to create tests such that safety varies greatly with a small fluctuation in T_safe, then the IQRs of T_safe provides a hint that when T_safe is configured between 0.94 and 2.36 seconds, Radius_tire between 0.32 and 0.35 meters, Power_max between 90.73 and 121.84 kW, etc, it is likely that the safety varies drastically when T_safe fluctuates.

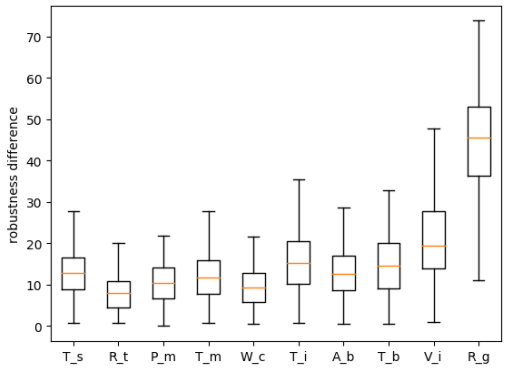
We remark that the implication of the IQRs is a rough distribution of the unstable region. On the other hand, the reported pairs of products in Table 2 are selected to show the smallest variation which leads to a safe-unsafe robustness transition. As it is an extreme case, some selected products happened not to fall in the IQRs.

6.3.2 Benefit for quality engineering. We also apply quartile analysis to configuration and robustness difference of the unstable individuals mentioned above. The resulting boxplots are shown in Figure 2. Observe that the configuration difference of R_gear is greater than other parameters due its discrete nature.

The information of the boxplots can be beneficial to the quality engineering of the model. For example, suppose a designer would like to decide how precisely the active safety feature should shut down the engine. The precision of the active safety feature reflects the required quality in the manufacturing process. The boxplots of T_safe show that half of the unstable configurations of T_safe have a configuration difference ranging roughly from 6.4% to 16.6%,



(a) Configuration difference



(b) Robustness difference

Figure 2: Boxplots of configuration and robustness difference for each parameter

and will cause a 8.9 to 16.5 difference in robustness. This information might help the designer to make a decision for the quality engineering of the active safety feature.

We discuss the problem setting and the results with the industrial partner who provides the model. They agree with our claim on the usefulness of the approach. They also confirm the validity of most results but find some points of surprise or gaps from what they expect according to their understanding. These points are marked for further investigation.

7 THREATS TO VALIDITY

We here discuss some threats that may affect the validity of the approach [23].

One common conclusion validity threat with the use of randomized algorithms is their inherent randomness. To deal with such randomness, we repeat our experiments 30 times for each parameter. Followed by this, we compare 30 runs of NSGA-II with the 30 runs of RS using Mann-Whitney U Test and A12 statistics based on the published guidelines [5].

Regarding construct validity threat, we choose HV to compare NSGA-II and RS to compare the quality of solutions produced by these algorithms. HV is chosen based on the guidelines published in [22]. In addition, to have a fair comparison between NSGA-II and RS, we used the same stopping criterion for both algorithms, i.e., the number of fitness evaluations.

An internal validity threat is the selection of NSGA-II. However, this is the most commonly used multi-objective algorithm in SBSE. Also, the use of the default parameter settings for NSGA-II is another internal validity threat. However, the evidence has shown that default parameters give good results [6].

The main external validity threat regards the use of only one case study. However, note that we experiment it with 10 parameters. We acknowledge that in the future more case studies are needed to generalize the results about the effectiveness of the approach.

8 RELATED WORK

Different notions of multi-PL exist in the literature, and interested readers may consult the following literature for details [15]. Here, we present related work from the following perspectives: 1) Safety Analyses (e.g., verification and testing) on Software Product Lines (SPLs), 2) Testing of Advanced Driving Assistance Systems (ADASs), 3) Sensitivity Analyses.

Safety Analyses on SPLs. There exist works focusing on verification and testing of SPLs. Some notable works include verification of Java SPLs [8], JML specifications [20], and testing SPLs with input-output featured transition systems [7]. However, there is a need for work related to hybrid systems and falsification. Also, the existing works do not focus on multi-PL level safety analyses, which is the focus of this paper. In general, there exist works on verification and testing of SPLs, but we propose to extend the notion of these analyses to the family of configurations and, in this paper, we demonstrate this for stability analysis in the automotive domain.

Testing ADASs. There are some recent works on testing ADASs that relate to our work such as by Abdessalem et al. [1–3]. There are two key differences. First, these works focus on testing, whereas we focus on stability analysis. Second, these works focus on one configuration, whereas we focus at the multi-PL level, i.e., stability analysis across families of cars.

Sensitivity Analyses. Works related to sensitivity analyses [18] are closer to our work. For example, Matlab supports sensitivity analysis, which is similar to our work in the sense that it focuses on studying how system behavior is affected by small variations in the values of system parameters. Matlab focuses on random/predefined points and performs sensitivity analysis closer to those points. Instead, we focus on systematically exploring the search space of system parameters with carefully designed objectives to perform stability analysis across multi-PL. Note that Matlab focuses on one configuration at a time, whereas we find a pair of configurations that are very similar to each other in terms of parameter values, but one leads to a safety violation and the other doesn't.

Another related work is by Harman et. al [13] that focuses on sensitivity analysis for the next release problem (NRP), i.e., the problem of selecting requirements to be implemented in the next release of software, while satisfying various objectives, e.g., customer satisfaction, and financial constraints. The key idea is to use search algorithms to assist a requirement engineer to perform a sensitivity analysis for the cost of implementing requirements. This assistance is essential since estimating the cost of implementing requirements is often error-prone. The approach is based on using different versions of the development costs and assessing the significance of the modified cost on the overall development costs. Our work is similar to this work in the sense that we also study the small variations in system parameters that lead to safety violations, whereas they study slight variations in the development costs for the NRP. Also, like our work, they use search algorithms; however, the search objectives and the problem to be solved are different.

9 CONCLUSIONS AND FUTURE WORK

The paper presents a technique to discover instability of safety in an automotive hybrid system. It relies on a multi-objective optimization problem to characterize instability, and adopts a search-based approach to solve the problem. The technique has been experimented on an industrial Simulink model of a safety feature in a car, and successfully detected instability of the model.

For future works, we plan to investigate the applicability of the technique to various configurable systems, especially models with time-dependent input signals. This extension would make the search space much more large, as the time-dependent input signals also need to be encoded in an individual. Second, in the current approach, we only look for pairs of configurations in which only one parameter changes. Although this is important to assess the effect of a single parameter, we might overlook the case where a parameter P_i can affect the overall safety only when changed together with another parameter P_j (i.e., P_i and P_j are interacting [19]). Therefore, we plan to extend the technique to assess instability caused by more than one parameter.

ACKNOWLEDGMENTS

N.-Z. Lee, P. Arcaini, and F. Ishikawa are supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST. S. Ali is supported by the Zen-Configurator project (grant no. 240024) funded by Research Council of Norway. We thank our industrial partner for the fruitful discussions.

REFERENCES

[1] Raja B. Abdessalem, Shiva Nejati, Lionel C. Briand, and Thomas Stifter. 2016. Testing Advanced Driver Assistance Systems Using Multi-Objective Search and Neural Networks. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering (ASE'16)*. ACM, 63–74.

[2] Raja B. Abdessalem, Shiva Nejati, Lionel C. Briand, and Thomas Stifter. 2018. Testing Vision-Based Control Systems Using Learnable Evolutionary Algorithms. In *Proceedings of the 40th International Conference on Software Engineering (ICSE'18)*. ACM, 1016–1026.

[3] Raja B. Abdessalem, Annibale Panichella, Shiva Nejati, Lionel C. Briand, and Thomas Stifter. 2018. Testing Autonomous Cars for Feature Interaction Failures Using Many-Objective Search. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE'18)*. ACM, 143–154.

[4] Shaukat Ali, Paolo Arcaini, Ichiro Hasuo, Fuyuki Ishikawa, and Nian-Ze Lee. 2019. Towards a Framework for the Analysis of Multi-Product Lines in the Automotive Domain. In *Proceedings of the 13th International Workshop on Variability Modelling of Software-Intensive Systems (VAMOS '19)*. ACM, New York, NY, USA, Article 12, 6 pages. https://doi.org/10.1145/3302333.3302345

[5] Andrea Arcuri and Lionel C. Briand. 2011. 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'11)*. ACM, 1–10.

[6] Andrea Arcuri and Gordon Fraser. 2011. On Parameter Tuning in Search Based Software Engineering. In *Proceedings of the 3rd International Conference on Search Based Software Engineering (SSBSE'11)*. Springer, 33–47.

[7] Harsh Beohar and Mohammad R. Mousavi. 2016. Input-Output Conformance Testing for Software Product Lines. *Journal of Logical and Algebraic Methods in Programming* 85, 6 (2016), 1131–1153.

[8] Daniel Bruns, Vladimir Klebanov, and Ina Schaefer. 2010. Verification of Software Product Lines with Delta-Oriented Slicing. In *Proceedings of the 1st International Conference on Formal Verification of Object-Oriented Software (FoVeOOS'10)*. Springer, 61–75.

[9] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. A. M. T. Meyarivan. 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197.

[10] Alexandre Donzé and Oded Maler. 2010. Robust Satisfaction of Temporal Logic over Real-Valued Signals. In *Proceedings of the 8th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'10)*. Springer, 92–106.

[11] Juan J. Durillo and Antonio J. Nebro. 2011. jMetal: A Java Framework for Multi-Objective Optimization. *Advances in Engineering Software* 42, 10 (2011), 760–771.

[12] Zhun Fan, Yi Fang, Wenji Li, Jiewei Lu, Xinye Cai, and Caimin Wei. 2017. A Comparative Study of Constrained Multi-Objective Evolutionary Algorithms on Constrained Multi-Objective Optimization Problems. In *2017 IEEE Congress on Evolutionary Computation (CEC'17)*. IEEE, 209–216.

[13] Mark Harman, Jens Krinke, Jian Ren, and Shin Yoo. 2009. Search Based Data Sensitivity Analysis Applied to Requirement Engineering. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO'09)*. ACM, 1681–1688.

[14] Mark Harman, Afshin Mansouri, and Yuanyuan Zhang. 2012. Search-Based Software Engineering: Trends, Techniques and Applications. *ACM Computing Surveys (CSUR)* 45, 1 (2012), 11.

[15] Gerald Holl, Paul Grünbacher, and Rick Rabiser. 2012. A Systematic Review and An Expert Survey on Capabilities Supporting Multi Product Lines. *Information and Software Technology* 54, 8 (2012), 828–852.

[16] James Kapinski, Jyotirmoy V. Deshmukh, Xiaoqing Jin, Hisahiro Ito, and Ken Butts. 2016. Simulation-Based Approaches for Verification of Embedded Control Systems: An Overview of Traditional and Advanced Modeling, Testing, and Verification Techniques. *IEEE Control Systems* 36, 6 (2016), 45–64.

[17] Antonio J. Nebro, Juan J. Durillo, and Matthieu Vergne. 2015. Redesigning the jMetal Multi-Objective Optimization Framework. In *Proceedings of the Companion Publication of the 17th Annual Conference on Genetic and Evolutionary Computation (GECCO Companion'15)*. ACM, 1093–1100.

[18] Andrea Saltelli, Karen Chan, and Marian Scott. 2000. *Sensitivity Analysis*. Vol. 1. Wiley New York.

[19] Larissa R. Soares, Pierre-Yves Schobbens, Ivan do Carmo Machado, and Eduardo S. de Almeida. 2018. Feature Interaction in Software Product Line Engineering: A Systematic Mapping Study. *Information and Software Technology* 98 (2018), 44–58.

[20] Thomas Thüm, Ina Schaefer, Sven Apel, and Martin Hentschel. 2012. Family-Based Deductive Verification of Software Product Lines. In *Proceedings of the 11th International Conference on Generative Programming and Component Engineering (GPCE'12)*. ACM, 11–20.

[21] Frank J. van der Linden, Klaus Schmid, and Eelco Rommes. 2007. *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. Springer.

[22] Shuai Wang, Shaukat Ali, Tao Yue, Yan Li, and Marius Liaaen. 2016. A Practical Guide to Select Quality Indicators for Assessing Pareto-Based Search Algorithms in Search-Based Software Engineering. In *Proceedings of the 38th International Conference on Software Engineering (ICSE'16)*. IEEE, 631–642.

[23] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering*. Springer Science & Business Media.