

Article

Adaptive Test Suits Generation for Self-Adaptive Systems Using SPEA2 Algorithm

Muhammad Abid Jamil ^{1,*}, Mohamed K. Nour ¹, Saud S. Alotaibi ², Mohammad Javed Hussain ²,
Syed Mutiullah Hussaini ¹ and Atif Naseer ³

¹ Department of Computer Science, College of Computers and Information Systems, Umm Al Qura University, Makkah 21955, Saudi Arabia; mknour@uqu.edu.sa (M.K.N.); sshussaini@uqu.edu.sa (S.M.H.)

² Department of Information Systems, College of Computers and Information Systems, Umm Al Qura University, Makkah 21955, Saudi Arabia; ssotaibi@uqu.edu.sa (S.S.A.); mjhussain@uqu.edu.sa (M.J.H.)

³ Science and Technology Unit, Umm Al Qura University, Makkah 21955, Saudi Arabia; anahmed@uqu.edu.sa

* Correspondence: majamil@uqu.edu.sa

Abstract: Self-adaptive systems are capable of reconfiguring themselves while in use to reduce the risks forced by environments for which they may not have been specifically designed. Runtime validation techniques are required because complex self-adaptive systems must consistently offer acceptable behavior for important services. The runtime testing can offer further confidence that a self-adaptive system will continue to act as intended even when operating in unknowable circumstances. This article introduces an evolutionary framework that supports adaptive testing for self-adaptive systems. The objective is to ensure that the adaptive systems continue to operate following its requirements and that both test plans and test cases continuously stay relevant to shifting operational conditions. The proposed approach using the Strength Pareto Evolutionary Algorithm 2 (SPEA2) algorithm facilitates both the execution and adaptation of runtime testing operations.

Keywords: search-based software engineering; adaptive systems; configurable systems; multi-objective evolutionary algorithms

Citation: Jamil, M.A.; Nour, M.K.; Alotaibi, S.S.; Hussain, M.J.; Hussaini, S.M.; Naseer, A. Adaptive Test Suits Generation for Self-Adaptive Systems Using SPEA2 Algorithm. *Appl. Sci.* **2023**, *13*, 11324. <https://doi.org/10.3390/app132011324>

Academic Editor: Vito Conforti

Received: 6 August 2023

Revised: 7 October 2023

Accepted: 9 October 2023

Published: 15 October 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

When software systems are subjected to changing environments, the open-loop structure that traditionally guides software development necessitates human monitoring [1]. Software systems are designed with feedback loops so they may adjust on their own to changing environments, reducing the need for human supervision. These closed-loop systems are known as self-adaptive software systems [2] and they are frequently divided into two components: an adaptation engine that realizes the feedback loops and controls the adaptable software [1]. Such systems will be crucial for safety-critical applications, where they must meet stringent criteria, as mentioned by Calinescu [3].

Some strategies cover self-adaptive system testing, but only for later development phases [4] through [5], after the systems have been put into use. It is not recommended to test the feedback loop in its entirety. Contrarily, we view the testing of self-adaptive system components as a prerequisite to early validation because a system with fully integrated adaptive software and a feedback loop is only available in the most advanced stages of development [6].

There are two major obstacles to utilizing search algorithms to optimize the self-adaptive system at runtime: First of all, it is challenging to successfully adapt with suitable feature encoding in the representation of optimization. The context of a search algorithm

may be applied to the SAS design. Managing dependencies between the features is very challenging, e.g., changing cache mode requires that the cache feature be turned on. The introduction of numerical features might make dependencies even more complex. For instance, in the Tomcat server, the number of maximum threads should not be smaller than the size of the minimum spare threads [7]. This is because of SAS's complex design and the inability of most search algorithms to handle dependency restrictions in nature. Secondly, it is difficult and complex to manage the tradeoffs between several competing objectives, particularly for SAS dynamically. This is explained by the abundance of potential adaptation strategies and the requirement that the chosen strategy works. The contradictory relationships between objectives are further complicated by the dynamic and variable character of SAS, making it challenging to investigate the tradeoff relationship. If these issues are not properly resolved, the quality of the SAS runtime optimization may be degraded, the running overhead may be undesirable, and the tradeoffs may be unbalanced [8].

Regarding the first challenge, as mentioned above, the conversion of the SAS design into the context of the search algorithm was performed manually [2,9,10], which makes the process costly. Furthermore, feature dependencies are frequently disregarded, wasting important function evaluation time on incorrect solutions during SAS runtime with little assurance of finding accurate solutions. In order to improve SAS during runtime, researchers [11,12] have linked the feature model [13] with search techniques, while taking category relationships into account. The use of search techniques to solve issues with the software product line would be the inspiration for this study [14]. The method frequently uses a straightforward binary format to encode all the features. This could result from the diverse dimensions and add needless complexity to the SAS at a dynamic level. For the second problem, SAS runtime efficiency has been achieved by utilizing exact search [11,15] and objectives aggregation as a weighted sum methodology. Modern SAS, however, frequently demonstrates considerable variability, expanding the search space of all potential solutions and making the issue unmanageable. From this point on, the precise search may runtime fail to scale. Search-based software engineering (SBSE) typically uses stochastic search, especially when combined with evolutionary algorithms, on the other hand, tends to be inherently reliable when it comes to tackling issues with extremely huge numbers of different solutions [16].

When designed correctly, those algorithms can produce approximatively and almost optimum solutions for challenging software engineering issues in a manageable amount of time [17]. Stochastic search has also been shown to be successful for several real-time systems [10,18–20]. Existing methods frequently convert a multi-objective problem into an aggregated single-objective problem and then use a single-objective evolutionary algorithm to optimize SAS [9,20]. Although objective aggregation may be advantageous in some situations, it has been demonstrated that there are some situations where software developers find it difficult to assign weights to various objectives, making it difficult for the aggregate to retain a wide variety of solutions [16].

NSGA-II [20], a well-known multi-objective evolutionary algorithm (MOEA), has been used in studies [2,10,19] to optimize SAS without utilizing weighted aggregation. The researchers demonstrated that MOEA is capable of discovering convergent and varied solutions as alternatives to optimizing via objective aggregation that emerges in the tradeoff space.

However, the coarse diversity preservation method of the NSGA-II prevents it from offering well-distributed solutions in some circumstances [21]. As a result, it would be ideal to have an optimized framework that can easily be used with many MOEAs to optimize SAS without being constrained by a single algorithm. Additionally, because MOEAs generate a variety of non-dominated solutions, the SAS is at risk of making unfair compromises because there is no proven technique for selecting the best one for adaptation at runtime [8].

2. Related Work

Some state-of-the-art works have been discussed in this section, which becomes a motivation for our proposed approach.

By storing copies on servers at physically distant locations, remote data mirroring (RDM) is a data security approach to ensure data availability and reduce data loss [22,23]. The RDM network must also efficiently replicate and disseminate data by minimizing bandwidth consumption and offering assurance that dispersed data are not lost or corrupted. In reaction to uncertainties, such as dropped or delayed messages and network link failures, the RDM can be reconfigured in real-time. Each network link also has a measurable throughput, latency, and loss rate, as well as operational costs that have an impact on a controlling budget. These indicators are used to assess the overall effectiveness and dependability of the RDM. The RDM can change its network topology and data mirroring algorithms to address unforeseen problems. The RDM application can be described and executed as a SAS because of its sophisticated and adaptable nature [24].

Another approach is adopted by [8], where using the elitist chromosomal representation to encode a problem into MOEA can improve quality and reduce runtime overhead for SAS optimization. Such a gain is more likely to be modest when there are few viable solutions. The search may be adequately guided by the dependency-aware operators, who will discover solutions with greater convergence and diversity, producing better SAS optimization. Nevertheless, using dependency-aware operators without making sure the selected adaptation method is balanced may reduce its efficacy. The FEMOSAA approach uses different MOEAs to generate better quality for SAS runtime optimization compared to the leading frameworks in the literature. Comparing FEMOSAA to modern frameworks, it has a very low runtime overhead overall. Additionally, the additional work required for knee selection and dependency-aware operators is minimal; on occasion, they can even marginally shorten the MOEA runtime [8].

Proteus methodology is a runtime testing management based on requirements. Proteus is a framework specifically designed to execute testing tasks at runtime, such as test execution and online adaptation. Proteus offers two layers of test adaptation to achieve this: test case parameter value adaptation and test suite adaptation. The management of runtime testing activities, such as the modification and execution of test cases and test suites, is handled by the Proteus framework. Proteus carries out two primary jobs to aid in this strategy. Each SAS configuration has an adaptive test plan specified at design time, where each adaptive test plan consists of numerous test suites and each SAS configuration corresponds to a specific operational scenario. Second, each time a new SAS configuration is run, Proteus conducts a testing cycle that may include numerous iterations that each run a distinct test suite. Next, each task is outlined individually [25].

Our proposed approach can affect SAS and evolutionary algorithms, incorporating the best aspects of both disciplines. Researchers can use MOEAs to handle SAS optimization without having substantial prior MOEA experience, especially with the SPEA2 method. The automatic conversion of the feature model into an MOEA context can establish systematic and understandable domain knowledge for evolutionary computation researchers, who would be able to generate more novel results in terms of the SAS domain. As opposed to many search-based software engineering problems, our comprehensive methodology extends the internal structure of MOEA by automatically and dynamically retrieving SAS domain knowledge. Future work will expand SPEA2-SAS to manage additional competing objectives and apply it to other SAS domains.

3. Proposed Approach-SPEA2-SAS

3.1. Optimizing Test Cases Generation for Large Adaptive Systems

The SPEA2-SAS approach is based on a feature model for handling runtime testing. SPEA2-SAS is a framework specifically designed for running testing operations. A frame-

work for handling runtime testing activities, such as the adaptation in terms of SAS feature model configuration and execution of test cases and test suites, is called SPEA2-SAS. SPEA2-SAS carries out two primary jobs to aid in this strategy. Each SAS configuration has an adaptive test strategy specified at design time, where each adaptive test strategy consists of numerous test suites and each SAS configuration corresponds to a specific operational scenario. Second, each time a new SAS configuration is run, SPEA2-SAS conducts a testing cycle that may include numerous iterations that each run a distinct test suite. An adaptive test strategy includes every test suite that might be created for a specific operating environment or set of system and environmental characteristics. To do this, for each SAS setup, SPEA2-SAS creates an adaptive test strategy that consists of a default test suite and several intermediate, automatically created test suites. We also need to discuss how the intermediate test suites are derived at runtime as well as how the default test suites are derived at design time.

For illustration purposes, consider that a SAS test engineer is setting up an adaptive test strategy for a certain SAS configuration in response to a specific set of operational conditions. In our case, two objectives need to be optimized: (1) testing cost and (2) the number of test cases that need to be optimized using the SPEA2 algorithm. There will be two types of test cases, valid tests and invalid tests, depending on the valid and invalid configurations of the feature model as a SAS configuration. To achieve optimized test adaptation, the SPEA2-SAS approach dynamically develops test suites at runtime. A new optimized test suite is generated based on the SAT solver methodology adopted in our research work. The SAT solver approach has been mentioned in this research work, which helps to select the features, and the SPEA2-SAS optimizer used a recursive loop that searched for feature combinations dynamically. Due to the continuous dynamic environment, the encoded recursive loop continuously adapts the SAS in order to enhance quality in terms of two-objective optimization. The adaptation process monitors the SAS and parameter values then finally updates the objective functions. The proposed selection searches for the best-balanced solution for adaptation in the form of non-dominated solutions. We consider a solution's execution order as a separate concern from optimization. We thus expect that, given a legitimate and optimal solution, the proposed method will enforce the proper order of execution by assessing the dependencies in the feature model.

If any fault occurs in terms of invalid test suits, the SAS infrastructure should reconfigure the feature model system and launch a fresh testing cycle with the reactivation of the valid test cases. The SAS can change features at runtime to optimize for different non-functional quality parameters like response time, cost, etc. SAS is frequently built to dynamically search for the feature combinations that lead to the best solutions to accomplish this aim and the adoption of search algorithms.

Figure 1 shows the SPEA2-SAS architecture with two main components: (1) describes the configuration scenario that creates valid configurations, (2) responsible for an adaptable scenario based on an evolutionary algorithm with the capacity to adapt feature combinations at runtime to optimize for various nonfunctional quality metrics, such as reaction time and cost.

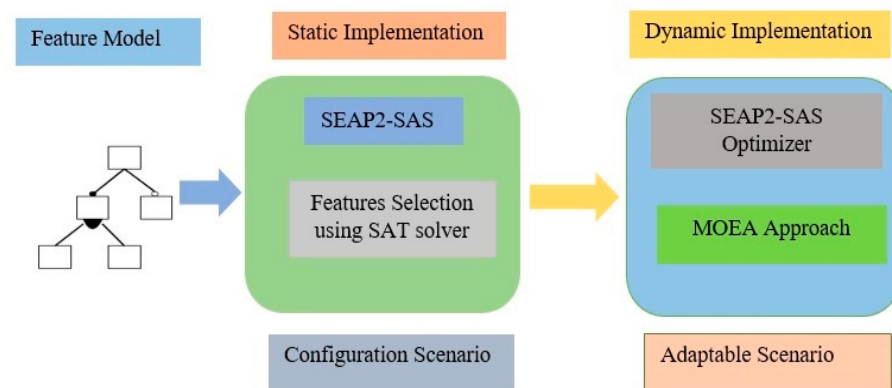


Figure 1. SPEA2-SAS based architecture.

3.2. Transforming Feature Model of SAS to Evolutionary Algorithm (SPEA2)

It is challenging to select the appropriate features and encode them into the representation of the search algorithm in a complicated SAS that has numerous features and settings. A potential search algorithm's capacity to search may therefore be positively or negatively impacted by the encoding of features. Such a representation creates the essential search space of the issue that has to be investigated in order to improve the SAS during runtime. Existing work [11, 12] merely encodes each feature in binary form and is redundant since certain elements in the SAS design might reflect the same aspect of variability [26] or do not contribute to the SAS's variability.

3.3. SAS Design Dependencies and Conflicting Objectives

Numerous popular stochastic and exact search algorithms (like MOEA) are not built to handle dependence constraints. Because of this, handling dependencies can be challenging, especially when there are a variety of categorical relationships included in the SAS dataset. If the constraints are handled incorrectly, then there are chances of degradation in adaptation quality [12]. Modern SAS frequently exhibits considerable unpredictability in the generation of a large number of configurations, which causes the search space to explode. To achieve numerous conflicting quality objectives in SAS at once, choices must frequently be made. The relative relevance of objectives may generally be accurately measured as numerical weights, which is challenging in some circumstances [16], according to several existing methodologies [9]. When such weights are improperly stated and represented, they invariably have a detrimental effect on the search process and lead to undesirable, subpar adaptation quality. Establishing a balanced compromise between objectives is considerably harder. These issues serve as the driving force behind our proposed approach, which automatically combines a particular feature model with a MOEA.

3.4. SPEA2 Approach for Optimizing the Objectives

SPEA2 makes use of a near-neighborhood density estimate approach, allowing it to direct its search strategy. SPEA2 employs the fitness assignment method, which takes into account each individual and how they impact or control others and vice versa [27]. In their research work, Gueorguiev et al. used the multi-objective genetic algorithm (MOGA) SPEA2 to solve time and robustness problems in software project planning with better Pareto fronts for positive results [28].

Figure 2 explains the pseudocode of the SPEA2 algorithm, where fitness criteria for each individual are described in the population.

Procedure SPEA2	
Input: $N', g, f_k(X) \triangleright N'$ members evolved g generations to solve $f_k(X)$	
1	Initialize Population P' ;
2	Create empty external set E' ;
3	for $i = 1$ to g do
4	Compute the fitness of each individual in P' and E' ;
5	Copy all non-dominated individual from P' and E' to E' ;
6	Use the truncation operator to remove elements from E' when the capacity of the file has been exceeded;
7	If the capacity of E' has not been exceeded then use dominated individuals in P' to fill E' ;
8	Perform binary tournament selection;
9	Apply crossover and mutation;
10	end

Figure 2. Pseudocode of SPEA2 [29].

Figure 3 shows the SPEA2 algorithm workflow according to our proposed problem with two objectives (cost and time) that need to be optimized. First of all, the proposed procedure is initialized with a large configurable SAS. Then, the two objectives' fitness functions are defined. These fitness functions use parameter values, and these values are mentioned in section 4. The procedure works until optimized results are generated. However, if the stop condition is not fulfilled, then the procedure goes back to the mating criteria and variation process, and these values are given to fitness functions, and the remaining procedure works in the same manner to generate results.

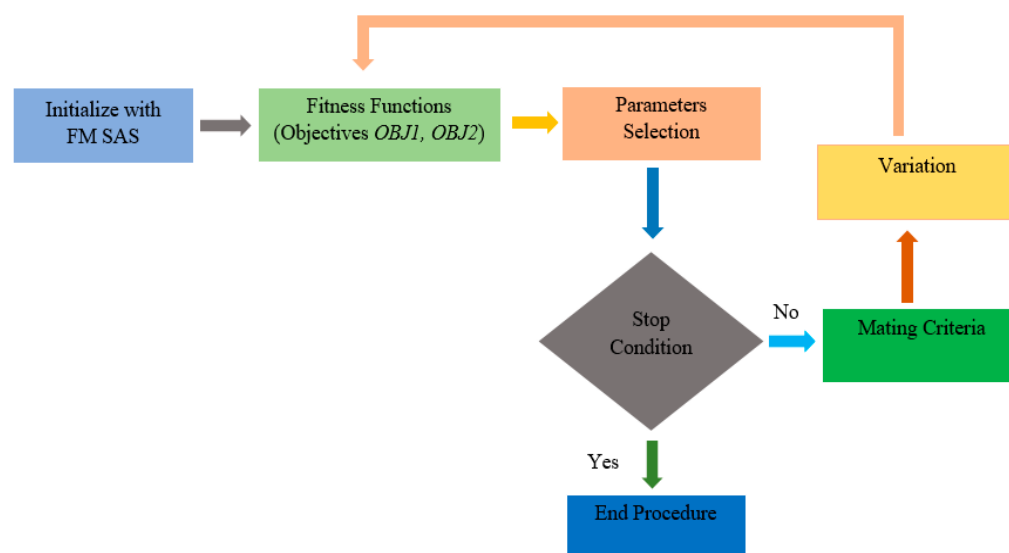


Figure 3. SPEA2 procedure workflow.

4. Experiments and Results

Objective Functions

The objectives (fitness) function as input utilized in the optimization is also defined with assistance from the elitist chromosome representation as mentioned above in the previous sections.

Due to the fact that the framework itself does not rely on any assumptions on how those objectives are organized internally, it is important to note that SPEA2-SAS works with a variety of measurable quality objectives while being indifferent to how the actual

objective operates. It is also feasible to take into account more than two objectives with SPEA2-SAS, but we only do so in this article for the sake of a more understandable explanation because the basic idea behind multi-objective optimization holds optimization regardless of the number of objectives. As long as they are consistent with the genes described by SPEA2-SAS, the actual goal functions used by SPEA2-SAS can be constructed using a variety of modeling methodologies from the literature (e.g., machine learning-based [18,30,31], analytical [32], and simulation-based [33]).

The elicited dependency chains and combinations of the genes are smoothly fed into the mutation operation at runtime to stop the generation of invalid offspring while mutating the solutions. In this study, we base our analysis on the boundary mutation operator, in which each gene is affected according to a mutation rate. A gene's optional values are assigned at random utilizing the mutation rate. The mutation operator was selected for experimentation because (a) it is widely used operator and (b) it deals with optimization problems to pick a random value from predefined values in the form of a tree for a SAS feature model, which is especially appropriate for our proposed optimization problem regarding SAS systems. However, because the breach of dependence is avoided anytime a gene is altered, SPEA2-SAS may easily be modified to work with any additional operators that change the genes similarly to the mutation operator.

When exchanging elements of the solutions, it is vital to eliminate invalid offspring, just like during the mutation process. To do this, the supplied crossover operator in the MOEA can be utilized with the extracted dependency chains and the merged value trees. In our study, there is also dependency on the crossover method, which involves two genes from different parents that are located in the same place on the chromosome and may be switched depending on the crossover rate. These uniform crossover operators were selected because they reduce the issue of gene position, which makes it easier to moderate other SAS design constraints by making them less sensitive to the proximity of highly reliant genes (features) in the encoding. It is simple to adapt SPEA2-SAS to operate with any other operator like mutation or crossover, provided that each pair of switched genes is always at the same place in the encoding. This is because whenever a pair of genes is switched, the violation of dependency is averted.

The violation of dependence is, however, avoided anytime a pair of genes is switched; hence, it is simple to adapt the SPEA2-SAS approach to operate with any other operators such that every pair of replaced genes would always be in the same place in the encoding.

The following two objectives need to be optimized:

1. Minimizing the Test Cases (OBJ1)
2. Minimizing the Cost (OBJ2)

We performed extensive tests to assess the efficacy of the SPEA2-SAS approach by contrasting it with its variations and cutting-edge frameworks under various metrics utilizing the actual operating feature model SAS. We specifically employ hyper-volume and a population size of 500 for SPEA2. The volume of the goal space is presented using the hype volume measure, and Pareto front A dominates this space. To assess the quality of the spread and convergence of the non-dominated solution set, a hyper-volume quality indicator is used. The greater the performance of an algorithm, the higher the HV value will be. This metric generates a single value that aids in estimating solutions that are not dominated [34].

For our real-time tests, we set up two feature models of SAS that are adopted for experiments. The two separate SASs are intended to look at the universality and application of SPEA2-SAS across various parameters. The proposed approach assists in showing how SPEA2-SAS may be used to analyze various feature models, reliance designs, environmental variables, aspects of quality targets, and levels of objective differences. In general, the large number of constraints means the feature model SAS is under more pressure, which reduces the number of viable solutions and increases the complexity of the issue.

The objective is to continuously decrease the following two incompatible quality criteria in the form of two-objective optimization.

We believe this is the case because, even though SPEA2-SAS can direct the search for better solutions, the advantages of avoiding the exploration of invalid solutions, which could be highly unbalanced for the competing objectives, have been overshadowed by randomly choosing a solution for adaptation from the final non-dominated options. In our experiments, the parameters have been customized for runtime efficiency concerning quality and overhead, or they are standard values. SPEA2 used in the experiments is extended from the MOEA framework [21].

Table 1 shows the SAS feature models selected for the experimentation process. The two feature models have been selected: SmartHome 2.2 and Coche Ecologico; their attributes have been mentioned in detail in Table 1, like the total number of features, configurations, and number of pairs.

Table 1. Feature model SAS.

Feature Models	Features	Configurations	Number of Pairs
Smart Homev2.2	60	3.87×10^9	6189
Coche Ecologico	94	2.32×10^7	11,075

Table 2 describes the number of parameters that have been adopted in our experiments. Before starting the experiments, parameter settings are adopted. To have a consistent outcome and to compare the effectiveness of the algorithms used in experiments, the fixed parameters have been decided upon at the beginning. The population size, the maximum number of assessments, the size of FM, crossover, and mutation rates are the selected parameter values. The number of evaluation iterations for each FM and method is the specified benchmark to determine whether to discontinue the experiment process.

Table 2. Parameters used for SPEA2 algorithm.

Parameter	Values
Population Size	250
Number of Generations	500
Crossover Rate	60%
Mutation Rate	40%

Table 3 displays the feature models (FMs) chosen for the experiment in the first column, along with the methods, solutions, generation convergence, and elapsed time (in milliseconds) in the other columns. The duration of time that an algorithm takes to complete from beginning to end is known as the elapsed time. The amount of time is expressed in milliseconds.

Table 3. Number of solutions with elapsed time.

Feature Model	Algorithm	Pareto Fronts Solutions	Generation Convergence	Elapsed Time (Millisecond)
Smart Home v2.2	SPEA2	200	16	25,991
Coche Ecologico	SPEA2	211	15	76,827

The proposed approach generated solutions that maintain a balanced trade-off between all objectives. This is especially intriguing since SPEA2-SAS targets situations where the relative relevance of SAS's competing objectives is uncertain and their quantification is too challenging. These balanced solutions are probably to occur around the obvious curve, showing a good compromise of solutions in terms of two objectives, as we can see in Figure 4, which offers options that minimize both the number of test cases and cost. These Pareto front solutions, also known as the extreme solutions, are often the ones that have the worst outcomes for each individual target. Finding these solutions in a non-

dominated set is comparable to seeking the solution or solutions with the greatest general distance from the set's extreme solutions.

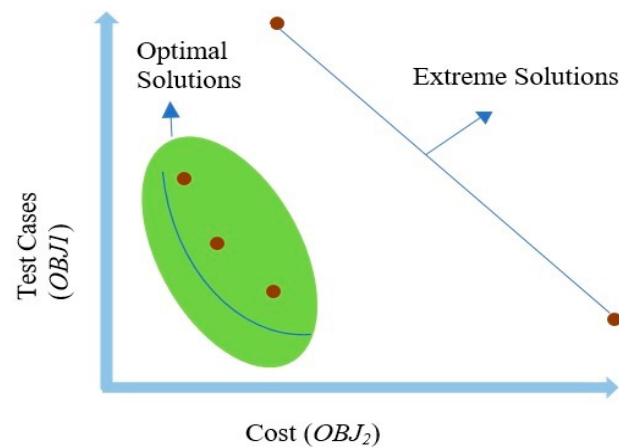


Figure 4. Optimal solutions Pareto front.

To accomplish this, we implement a SPEA2 technique for selecting just one solution from the MOEA results set. We are given the final non-dominated set produced by the feature-guided MOEA, as illustrated in Figure 4. SPEA2-SAS may operate with a variety of MOEAs with no loss of generality. In this study, we use one MOEA to run SPEA2-SAS.

We utilized these feature models by keeping track of the SAS data under a random workload for 250 intervals to balance the objective functions in this study, and then the feature models are progressively and dynamically modified during runtime. On the feature model, we also included several categories and numerical relationships. At each iteration, we modified the throughput and cost of particular concrete services step by step by altering their variety level following the hypervolume measure, simulating dynamism and unpredictability under time-varying environmental circumstances. More varied concrete feature models typically indicate SAT solver adaptation solutions, which lowers the number of efficient solutions and hence makes the task more challenging. Here, we can evaluate maximizing and minimizing the following competing objectives for the total composition.

Additionally, for each interval, we contrast the percentage of appropriate adaption strategies discovered in the final population. It is unclear whether the elitist chromosome representation outperforms the traditional binary representation, as the aforementioned comparison only shows the elitist chromosome representation and dependency-aware operations provide good combinatorial performance and SAT solver selection. The main reason why elitist chromosome representation outperforms traditional binary representation is that it fundamentally reduces the search space without affecting SAS's original variability, increasing the likelihood that MOEA will find the best solutions, producing more valid solutions and higher-quality results.

5. Discussion

The quality of the searched valid solution determines how effective the proposed approach is. It is demonstrated in the above sections, when the search process wastes time exploring invalid solutions, the quality of the solutions in the final population may suffer, which would have a negative impact on the advantage provided by our proposed method SPEA2-SAS for the same reason.

5.1. State-of-the-Art Comparison

We contrast SPEA2-SAS with the following cutting-edge search-based frameworks from the literature to assess its efficacy further:

PLATO [9] is an optimization strategy that uses the famous single-objective evolutionary algorithm genetic algorithm together with a weighted sum of the objectives. Additionally, it does not take dependence into account while optimizing, thus we employ the same method as SPEA2-SAS. To identify the most effective solution, we assign the objectives equal weights. We chose elitist chromosome representations as PLATO relies on manual transposition to make comparisons that are fair and remove bias in repeatability.

FUSION [11] is a framework that uses SAS optimization as an integer programming issue using a model-based approach with an aggregate objective function that is solved using an exact algorithm (in the tests, we employ branch-and-bound). FUSION solely takes into account category dependence and employs a binary representation of the solutions. As a result, when there is no viable option, we correct the numerical dependence violations.

SPEA2-SAS achieves competitive results on runtime overhead in comparison to state-of-the-art frameworks, but the actual time required by SPEA2-SAS depends on the underlying MOEA. Notably, FUSION has the highest overhead because its exact algorithm cannot scale with a large search space, and the optimization runs are frequently forcibly returned. SPEA2-SAS's ability to develop synergy between SAS software engineering and evolutionary computing is its most prominent advantage. Software programmers are given the freedom to modify an MOEA's behaviors in whatever way they are accustomed to (i.e., the feature) without in-depth knowledge of evolutionary algorithms in general.

SPEA2-SAS makes use of MOEA [35], which is particularly appropriate in situations where the concerned weights of the objectives are uncertain and also challenging to define. We do admit that there are situations where the relative importance of competing goals is stated explicitly. In such instances, the strength of elitist chromosome representation and dependency-aware operators may still be utilized by the SAS. It is also important to note that MOEA does not ensure optimum solutions, but it is particularly effective at generating accurate approximations to difficult, nonlinear problems that would otherwise be impractical to solve by exact optimization. Therefore, we do not advocate utilizing the SPEA2-SAS approach for SAS that is straightforward, has a small search area, and can be resolved with an exact search that produces the best solutions.

To tackle our SAS optimization challenge, however, search-based software engineering methods, notably evolutionary algorithms, present a potential approach. This is because of the algorithms' dynamic nature, which allows them to undertake optimization without requiring in-depth knowledge of or presumptions about the issue at hand (such as the SAS's characteristics or the environment). Additionally, it is possible to develop roughly optimal solutions for even the most difficult situations because of the concepts of natural evolution and population.

5.2. Threats to Validity

The most frequently evaluated quality aspects of SAS from the literature [36,37] were chosen for our experiments in this work, including response time, throughput in terms of test cases, cost, etc. However, there are some questions about whether the metrics we used can truly reflect what we want to measure.

The stochastic nature of the SPEA2-SAS that was taken into consideration in the experiments may potentially imply a threat to the construct validity, which can influence the measurements. To accurately interpret the results analysis for stochastic search-based optimization, it is a fact that representative measurements for the different quality metrics were evaluated using hypervolume and space, which are often used metrics to assess the

quality of solutions to multi-objective optimization problems [21,38]. Indeed, as noted in Acuri and Briand [39], numerous runs are required to meaningfully interpret the data for stochastic search-based optimization. By employing the design outlined in Acuri and Briand [39], which included carrying out 500 optimization runs for two feature models of SAS, we were able to reduce this bias.

We also examined the commonly used but unique SPEA2 under two operating models of SAS and various parameter settings, which might vary the runtime behaviors of the SAS, to increase the universality of experimental evaluations. It is challenging to claim total generality even though our testing on the instances approximates real and industrial size; such a demand would need a considerably larger number of independent domain-specific examples and needs to be carried out by software developers or testers.

6. Conclusions

To optimize SAS at runtime, this article introduces SPEA2-SAS, a unique framework that automatically combines a SAS feature model with an MOEA in a systematic way. To build a chromosome representation, SPEA2-SAS identifies elitism characteristics at design time, including categorical and numerical ones. It then extracts gene dependencies, which are subsequently utilized to expand the underlying MOEA for runtime optimization. The engineers' domain expertise, the feature model, may narrow the search space and guide the search, boosting the likelihood of coming up with better solutions.

SPEA2-SAS further identifies optimal solutions that produce a balanced tradeoff between two objectives. We also demonstrate how SPEA2-SAS produces better and more balanced results for tradeoffs with reasonable overhead by thoroughly contrasting it with its variants and cutting-edge frameworks on two complex real-world SAS with different parameters like mutation, crossover values, etc. In particular, the most noteworthy findings of SPEA2-SAS include the following; (1) While using the elitist chromosome representation to encode an issue into MOEA can enhance quality and reduce runtime overhead for SAS optimization, such a gain is more likely to be modest when there are few viable options, like when the workload is high in terms of a large number of features of a SAS, (2) The search may be adequately guided by the dependency-aware operators, who will discover solutions with greater convergence and diversity metrics like hypervolume as mentioned in the above sections, producing superior SAS optimization. However, using dependency-aware operators without making sure the chosen adaptation method is balanced may reduce its efficacy, (3) The choice of SAT solver aids in locating a more sensible adaptation strategy in terms of a valid test case suite, (4) SPEA2-SAS, with the tested SPEA2, delivers statistically and practically better quality for SAS runtime optimization than the state-of-the-art frameworks from the literature.

The proposed work influences and develops the synergy between evolutionary computing and software engineering for SAS, incorporating the best aspects of both disciplines. Software developers may use MOEAs to solve SAS optimization without having substantial prior MOEA experience, especially with SPEA2-SAS. On the other hand, automating the feature model's conversion into an MOEA context can help to learn MOEA and create domain expertise understandably and systematically for academics studying MOEA, which can help in designing SAS algorithms that are more effective.

As opposed to many search-based software engineering (SBSE) problems, our deeper synergy extends the internal structure of MOEA by automatically and dynamically retrieving SAS domain knowledge. In subsequent work, we want to enhance SPEA2-SAS to manage additional competing objectives by applying it to more SAS domains.

We believe that combining techniques like the one we have described with approaches from a different category will have a similar positive impact on quality assurance for SASS. According to what we understand, this is especially true for SAS, where runtime surprises may occur and need to be properly handled. However, testing must still be carried out before a system is put into use to guarantee at least a foundational level

of SAS quality. The proposed testing method can be developed in many ways in the future. Instead, we may utilize coverage criteria and a formal model to automatically generate test inputs and oracles taking SAS and its environment's unpredictability into account.

Our proposed work has contributed to the multi-objective research area in the following ways. First, the main reason why elitist chromosome representation outperforms traditional binary representation is that it basically minimizes the size of the search space without affecting SAS's original variability, increasing the likelihood that the selected MOEA (SPEA2) will find the best solutions, which will result in more valid solutions and higher-quality solutions generated with a shorter running time of the MOEA.

Second, despite the fact that SPEA2-SAS can direct the search for better solutions, it has the advantage of avoiding the exploration of invalid solutions, which could be highly unbalanced for the competing objectives. However, a solution from the final non-dominated solutions is randomly selected for adaptation. Third, as compared to other frameworks mentioned above, the contribution of SPEA2-SAS is that it demonstrates how the use of selection, operators, and elitist representation may direct the MOEA to produce outputs of higher quality even when employing various underlying MOEAs.

In further work, we intend to assess SPEA2-SAS in other extreme situations with large SAS models and constrained computing resources. It is important to keep in mind that developing the tests and setting up the SAS are not simple tasks, and they may cost money. The authors want to extend their future work by utilizing the optimization strategies indicated in the research of other scholars [39–47].

Author Contributions: The first author contributed equally to the research and wrote the article. Methodology, M.K.N.; software, M.K.N.; validation, S.S.A. and S.M.H.; investigation, M.K.N.; resources, M.J.H.; data curation, A.N. and M.A.J.; writing—original draft, M.A.J.; writing—review and editing, M.A.J.; supervision, M.A.J.; funding acquisition, S.S.A. All authors have read and agreed to the published version of the manuscript.

Funding: The authors extend their appreciation to the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia, for funding this research work through the project number: IFP22UQU4320619DSR113.

Institutional Board Review Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data will be available upon reasonable request.

Acknowledgments: The authors extend their appreciation to the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number: IFP22UQU4320619DSR113

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ismail, S.; Shah, K.; Reza, H.; Marsh, R.; Grant, E. Toward management of uncertainty in self-adaptive software systems: IoT case study. *Computers* **2021**, *10*, 27.
2. Arcelli, D. Exploiting queuing networks to model and assess the performance of self-adaptive software systems: A survey. *Procedia Comput. Sci.* **2020**, *170*, 498–505.
3. Gerostathopoulos, I.; Vogel, T.; Weyns, D.; Lago, P. How do we evaluate self-adaptive software systems?: A ten-year perspective of SEAMS. In Proceedings of the 2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), Madrid, Spain, 18–24 May 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 59–70.
4. Han, D.S.; Yang, Q.L.; Xing, J.C.; Ma, G.L. Easymodel: A refinement-based modeling and verification approach for self-adaptive software. *J. Comput. Sci. Technol.* **2020**, *35*, 1016–1046.
5. Han, D.; Ma, G.; Cai, Y.; Wang, B.; Li, A. ProbaSAS: Modeling and Decision-Making Approach for Self-Adaptive Software Systems under Uncertainty. In Proceedings of the 2022 41st Chinese Control Conference (CCC), Hefei, China, 25–27 July 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 5871–5876.

6. Fedasyuk, D.; Lutsyk, I. Method of modification of self-adaptive software systems based on ontology. In Proceedings of the 2022 IEEE 16th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), Lviv-Slavske, Ukraine, 22–26 February 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 530–533.
7. Apache Software Foundation. Apache Tomcat. Available online: <http://tomcat.apache.org/> (accessed on 24 June 2022).
8. Chen, T.; Li, K.; Bahsoon, R.; Yao, X. FEMOSAA: Feature-guided and knee-driven multi-objective optimization for self-adaptive software. *ACM Trans. Softw. Eng. Methodol. (TOSEM)* **2018**, *27*, 1–50.
9. Ramirez, A.J.; Knoester, D.B.; Cheng, B.H.; McKinley, P.K. Plato: A genetic algorithm approach to run-time reconfiguration in autonomic computing systems. *Clust. Comput.* **2011**, *14*, 229–244. <https://doi.org/10.1007/s10586-010-0122-y>.
10. Yusoh, Z.I.M.; Tang, M. Composite SaaS placement and resource optimization in cloud computing using evolutionary algorithms. In Proceedings of the 2012 IEEE 5th International Conference on Cloud Computing (CLOUD), Honolulu, HI, USA, 24–29 June 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 590–597. <https://doi.org/10.1109/CLOUD.2012.61>.
11. Han, D.; Cai, Y.; Chen, W.; Cui, Z.; Li, A. Timed-SAS: Modeling and Analyzing the Time Behaviors of Self-Adaptive Software under Uncertainty. *Appl. Sci.* **2023**, *13*, 2018.
12. Sulaiman, R.A.; Jawawi, D.N.; Halim, S.A. Cost-effective test case generation with the hyper-heuristic for software product line testing. *Adv. Eng. Softw.* **2023**, *175*, 103335.
13. Zhao, X.; Li, X.; Dong, X. Research on Component-Based Embedded Spacecraft Software System Development. In Proceedings of the 2023 8th International Conference on Intelligent Computing and Signal Processing (ICSP), Xi'an, China, 23–25 September 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 248–252.
14. Horcas, J.M.; Pinto, M.; Fuentes, L. Empirical analysis of the tool support for software product lines. *Softw. Syst. Model.* **2023**, *22*, 377–414.
15. Cardellini, V.; Casalicchio, E.; Grassi, V.; Iannucci, S.; Presti, F.L.; Mirandola, R. MOSES: A framework for QoS driven runtime adaptation of service-oriented systems. *IEEE Trans. Softw. Eng.* **2012**, *38*, 1138–1159. <https://doi.org/10.1109/TSE.2011.68>.
16. Hameed, S.; Elsheikh, Y.; Azzeh, M. An optimized case-based software project effort estimation using genetic algorithm. *Inf. Softw. Technol.* **2023**, *153*, 107088.
17. Hierons, R.M.; Li, M.; Liu, X.; Segura, S.; Zheng, W. SIP: Optimal product selection from feature models using many-objective evolutionary optimization. *ACM Trans. Softw. Eng. Methodol. (TOSEM)* **2016**, *25*, 1–39.
18. Chen, T.; Bahsoon, R. Self-adaptive trade-off decision making for autoscaling cloud-based services. *IEEE Trans. Serv. Comput.* **2015**, *10*, 618–632.
19. El Kateb, D.; Fouquet, F.; Nain, G.; Meira, J.A.; Ackerman, M.; Le Traon, Y. Generic cloud platform multi-objective optimization leveraging models@ run. time. In Proceedings of the 29th Annual ACM Symposium on Applied Computing, Gyeongju, Republic of Korea, 24–28 March 2014; Association for Computing Machinery: New York, NY, USA, 2014; pp. 343–350.
20. Fredericks, E.M. Automatically hardening a self-adaptive system against uncertainty. In Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS'16), Austin, TX, USA, 16–17 May 2016; ACM: New York, NY, USA, 2016; pp. 16–27. <https://doi.org/10.1145/2897053.2897059>.
21. Zhang, Q.; Li, H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **2017**, *11*, 712–731. <https://doi.org/10.1109/TEVC.2007.892759>.
22. Fidalgo, C.G.; Sousa, M.; Mendes, D.; Dos Anjos, R.K.; Medeiros, D.; Singh, K.; Jorge, J. Manipulating avatars and gestures to improve remote collaboration. In Proceedings of the 2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR), Shanghai, China, 25–29 March 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 438–448.
23. Keeton, K.; Santos, C.A.; Beyer, D.; Chase, J.S.; Wilkes, J. Designing for Disasters. *FAST* **2004**, *4*, 59–62.
24. Ramirez, A.J.; Knoester, D.B.; Cheng, B.H.; McKinley, P.K. Applying genetic algorithms to decision making in autonomic computing systems. In Proceedings of the 6th International Conference on Autonomic Computing, Barcelona, Spain, 15–19 June 2009; Association for Computing Machinery: New York, NY, USA, 2009; pp. 97–106.
25. Fredericks, E.M.; Cheng, B.H. Automated generation of adaptive test plans for self-adaptive systems. In Proceedings of the 2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, Florence, Italy, 18–19 May 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 157–167.
26. Benavides, D.; Segura, S.; Ruiz-Cortés, A. Automated analysis of feature models 20 years later: A literature review. *Inf. Syst.* **2010**, *35*, 615–636.
27. Zitzler, E.; Laumanns, M.; Thiele, L. *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*; TIK Report, 103; 2001.
28. Gueorguiev, S.; Harman, M.; Antoniol, G. Software project planning for robustness and completion time in the presence of uncertainty using multi objective search based software engineering. In Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, Montreal, QC, Canada, 8–12 July 2009; Association for Computing Machinery: New York, NY, USA, 2009; pp. 1673–1680.
29. Coello, C.A.C. *Evolutionary Algorithms for Solving Multi-Objective Problems*; Springer: Berlin/Heidelberg, Germany, 2007.
30. Chen, T.; Bahsoon, R.; Yao, X. Online QoS modeling in the cloud: A hybrid and adaptive multi-learners approach. In Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, London, UK, 8–11 December 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 327–336.
31. Chen, T.; Bahsoon, R. Self-adaptive and online QoS modeling for cloud-based software services. *IEEE Trans. Softw. Eng.* **2017**, *43*, 453–475.

32. Roy, N.; Dubey, A.; Gokhale, A.; Dowdy, L. A capacity planning process for performance assurance of component-based distributed systems. In Proceedings of the 2nd ACM/SPEC International Conference on Performance Engineering, Delft, The Netherlands, 12–16 March 2016; Association for Computing Machinery: New York, NY, USA, 2011; pp. 259–270.
33. Fittkau, F.; Frey, S.; Hasselbring, W. CDOSim: Simulating cloud deployment options for software migration support. In Proceedings of the 2012 IEEE 6th International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), Trento, Italy, 24 September 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 37–46.
34. Zitzler, E.; Thiele, L. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Trans. Evol. Comput.* **1999**, *3*, 257–271.
35. Hadka, D. MOEA framework—a free and open source Java framework for multiobjective optimization. *Version 2* **2015**, *74*.
36. Mingay, S. Green IT: The new industry shock wave. *Gart. RAS Res. Note G* **2007**, 153703.
37. Wada, H.; Suzuki, J.; Yamano, Y.; Oba, K. E³: A multiobjective optimization framework for SLA-aware service composition. *IEEE Trans. Serv. Comput.* **2011**, *5*, 358–372.
38. Li, M.; Chen, T.; Yao, X. A critical review of a practical guide to select quality indicators for assessing pareto-based search algorithms in search-based software engineering: Essay on quality indicator selection for SBSE. In Proceedings of the 40th International Conference on Software Engineering, NIER Track, Gothenburg, Sweden, 27 May–3 June 2018; IEEE: Piscataway, NJ, USA; ACM: New York, NY, USA, 2018.
39. Arcuri, A.; Briand, L. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In Proceedings of the 2011 IEEE 33rd International Conference on Software Engineering (ICSE), Honolulu, HI, USA, 21–28 May 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 1–10.
40. Kader, M.A.; Zamli, K.Z.; Alkazemi, B.Y. An Experimental Study of a Fuzzy Adaptive Emperor Penguin Optimizer for Global Optimization Problem. *IEEE Access* **2022**, *10*, 116344–116374. <https://doi.org/10.1109/ACCESS.2022.3213805>.
41. Odili, J.B.; Noraziah, A.; Alkazemi, B.; Zarina, M. Stochastic process and tutorial of the African buffalo optimization. *Sci. Rep.* **2022**, *12*, 17319. <https://doi.org/10.1038/s41598-022-22242-9>.
42. Alsewari, A.A.; Zamli, K.Z.; Al-Kazemi, B. Generating t-way test suite in the presence of constraints. *J. Eng. Technol. (JET)* **2015**, *6*, 52–66.
43. Wazirali, R.; Alasmary, W.; Mahmoud, M.M.; Alhindi, A. An optimized steganography hiding capacity and imperceptibly using genetic algorithms. *IEEE Access* **2019**, *7*, 133496–133508. <https://doi.org/10.1109/ACCESS.2019.2941440>.
44. Ahmad, A. Optimizing Training Data Selection for Decision Trees using Genetic Algorithms. *Int. J. Comput. Sci. Netw. Secur. (IJCSNS)* **2020**, *20*, 84.
45. Jami, M.A.; Nour, M.K. Managing Software Testing Technical Debt Using Evolutionary Algorithms. *Comput. Mater. Contin.* **2022**, *73*.
46. Jami, M.A.; Nour, M.K.; Alotaibi, S.S.; Hussain, M.J.; Hussaini, S.M.; Naseer, A. Software Product Line Maintenance Using Multi-Objective Optimization Techniques. *Appl. Sci.* **2023**, *13*, 9010.
47. Jami, M.A.; Alsadie, D.; Nour, M.K.; Awang Abu Bakar, N.S. Maintain Optimal Configurations for Large Configurable Systems Using Multi-Objective Optimization. *Comput. Mater. Contin.* **2022**, *73*.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.