On the Interaction between the Search Parameters and the Nature of the Search Problems in Search-Based Model-Driven Engineering

Isis Roca[1], Jaime Font[1], Lorena Arcega[1], and Carlos Cetina[1]

[1]Universidad San Jorge

January 24, 2023

Abstract

The use of Search-Based software engineering to address Model-Driven Engineering activities (SBMDE) is becoming more popular. Many maintenance tasks can be reformulated as a search problem, and, when those tasks are applied to software models, the search strategy has to retrieve a model fragment. There are no studies on the influence of the search parameters when applied to software models. This paper evaluates the impact of different search parameter values on the performance of an evolutionary algorithm whose population is in the form of software models. Our study takes into account the nature of the model fragment location problems (MFLPs) in which the evolutionary algorithm is applied. The evaluation searches 1,895 MFLPs (characterized through five measures that define MFLPs) from two industrial case studies and uses 625 different combinations of search parameter values. The results show that the impact on the performance when varying the population size, the replacement percentage, or the crossover rate produces changes of around 30% in performance. With regard to the nature of the problems, the size of the search space has the largest impact. Search parameter values and the nature of the MFLPs influence the performance when applying an evolutionary algorithm to perform fragment location on models. Search parameter values have a greater effect on precision values, and the nature of the model fragment location problems has a greater effect on recall values. Our results should raise awareness of the relevance of the search parameters and the nature of the problems for the SBMDE community.

Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

On the Interaction between the Search Parameters and the Nature of the Search Problems in Search-Based Model-Driven Engineering†

Isis Roca*[1,2] | Jaime Font[1] | Lorena Arcega[1] | Carlos Cetina[1]

[1]SVIT Research Group, Universidad San Jorge, Zaragoza, Spain
[2]PROS Research Centre, Universitat Politècnica de València, Valencia, Spain

Correspondence
*Isis Roca, Email: iroca@usj.es

Summary

The use of Search-Based software engineering to address Model-Driven Engineering activities (SBMDE) is becoming more popular. Many maintenance tasks can be reformulated as a search problem, and, when those tasks are applied to software models, the search strategy has to retrieve a model fragment. There are no studies on the influence of the search parameters when applied to software models. This paper evaluates the impact of different search parameter values on the performance of an evolutionary algorithm whose population is in the form of software models. Our study takes into account the nature of the model fragment location problems (MFLPs) in which the evolutionary algorithm is applied. The evaluation searches 1,895 MFLPs (characterized through five measures that define MFLPs) from two industrial case studies and uses 625 different combinations of search parameter values. The results show that the impact on the performance when varying the population size, the replacement percentage, or the crossover rate produces changes of around 30% in performance. With regard to the nature of the problems, the size of the search space has the largest impact. Search parameter values and the nature of the MFLPs influence the performance when applying an evolutionary algorithm to perform fragment location on models. Search parameter values have a greater effect on precision values, and the nature of the model fragment location problems has a greater effect on recall values. Our results should raise awareness of the relevance of the search parameters and the nature of the problems for the SBMDE community.

KEYWORDS:
Model-Driven Engineering (MDE), Search-Based Software Engineering (SBSE), Search Parameters, Model Fragment Location, Evolutionary Algorithm (EA)

1 | INTRODUCTION

There is a growing body of research on the use of Search-Based Software Engineering (SBSE) to address Model-Driven Engineering (MDE) activities[1]. In the intersection between MDE[2] and SBSE[3], activities related to model maintenance are reformulated as search problems. Specifically, tasks such as bug localization, traceability link recovery, or feature location can

†This work has been partially supported by the Spanish Ministry of Science and Innovation under the project *Iniciando Lineas de Producto Software Mediante Busquedas Interactivas Dirigidas por Modelos* (PID2021-128695OB-I00).

be reformulated as search problems where a model fragment must be found[4,5,6]. These works use search-based optimization techniques (mainly those from the evolutionary computation literature) to automate the search for optimal and near-optimal solutions.

In this work, we apply Search-Based Model-Driven Engineering (SBMDE) to address location problems in models. This activity aims to identify the parts of the models that are relevant to a specific task, and the result comes in the form of a model fragment. Model fragment location is one of the most important search problems in models. For example, in a model with 500 elements, the number of model fragments that can be generated can reach the value of 10^{29} [7]. Since the search space is so large, it is not practical to thoroughly explore the space of possibilities.

In our previous works[8,9], we proposed an approach for locating model fragments using an evolutionary algorithm. To achieve this, the evolutionary algorithm keeps a population of candidate solutions (in the form of model fragments) and evolves them using genetic operators that are designed to work with model fragments.

However, in current SBMDE practice, key search parameter values (population size, replacements, mutation rate, and crossover rate) are selected by conventions and ad hoc choices. Nevertheless, the default values used by the SBMDE community[1] are borrowed from other domains, and there are no specific studies about what search parameter values should be used when working with software models.

In addition, works on SBMDE provide many details about the techniques that are used to perform the search, however there is a lack of detail about the nature of the problems used. The proper reporting of the nature of the problems used is important so that the results obtained can be useful to other practitioners. Therefore, we characterize the nature of problems by means of five measures that define model fragment location problems (size, volume, density, multiplicity, and dispersion)[10].

In this work, we perform an evaluation to determine which search parameter has the highest impact on the performance of the search based on the nature of the location problems. To do so, we apply the search strategy to 1,895 Model Fragment Location Problems (MFLP) from industrial case studies using different sets of search parameter values (625 different combinations). Then, we calculate the performance by means of precision, recall, and F-measure values[11], comparing the results provided in each case with the oracle extracted from the case studies (considered the ground truth). Finally, we perform a statistical analysis of the results obtained in order to determine whether the different search parameter values applied and the nature of the MFLPs have an impact on the performance or whether the differences are obtained by mere chance.

The results shows that the choice of good search parameters can provide a boost in performance (precision mean values range from 6.48% for non-optimal parameters to values up to 73.08% for optimal parameters, while mean recall values range from 62.02% up to 75.45%). Moreover, when varying the population size, replacement percentage, or crossover rate parameters, the differences in performance are around 30%, but, when varying the mutation rate parameter, the differences in performance remain below 5%. Our statistical analysis shows that the crossover rate parameter has the largest impact on our case studies on industrial models. In addition, the nature of the model fragment location problems also influences the precision and recall values. Low values in the measures related to the search space obtain better values in performance, while high values in measures related to the model fragment outperform the low values. In this case, the size of the search space has the greatest impact. It is important to highlight that while the search parameter values have a greater impact on precision values, the nature of the MFLPs has a greater impact on recall values.

The evaluation presented in this paper is an initial work on the optimization that can be achieved when applying an evolutionary algorithm to perform fragment location on models. We want to advise the SBMDE community that they could be missing out on the relevance of the search parameters and the nature of their case studies for SBMDE approaches. We present this evaluation with the hope that it can be useful for practitioners from the SBMDE community when looking for default values or advice on how to balance their search strategy to boost performance.

The rest of the paper is organized as follows. Section 2 presents some background about model fragment location using an evolutionary algorithm. Section 3 shows the process that is followed to evaluate the impact of the different parameter values taking into account the measures for reporting model fragment location problems. Section 4 presents and discusses the results of the evaluation. Section 5 examines some related work. Finally, we conclude the paper in Section 6.

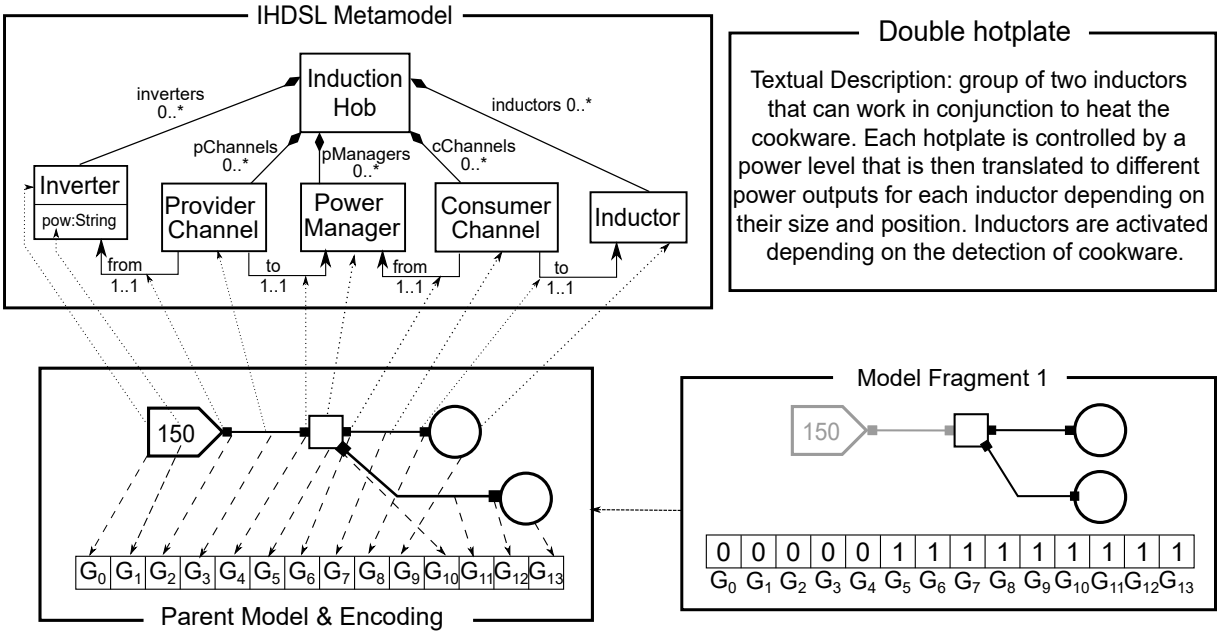


FIGURE 1 IHDSL metamodel, product model, and model fragment.

2 | MODEL FRAGMENT LOCATION USING AN EVOLUTIONARY ALGORITHM

In this section, we present the model fragment location problem based on the products of one of our industrial partners, the search strategy used to address the model fragment location problems, the measures that define model fragment location problems, and the search parameters of the evolutionary algorithm.

2.1 | Model fragment location problem (MFLP)

Figure 1 shows the Domain-Specific Language (DSL) used by one of our industrial partners, BSH, to formalize its products. This DSL is used to describe the models of the induction hobs that will be part of the evaluation. The firmware of the BSH products is generated from the DSL models. The DSL used by our industrial partner to specify the Induction Hobs (IHDSL) is composed of 46 metaclasses, 47 relations with each other, and more than 180 properties. For legibility reasons and due to intellectual property rights concerns, in this section, we show a simplified subset of the IHDSL (see Figure 1, IHDSL Metamodel). However, the evaluation was performed using the full IHDSL that is used in BSH.

The Parent Model in Figure 1 depicts an example of a model that is specified with the IHDSL. In this example, the induction hob aggregates a 150-power inverter that is connected to a single power manager through a provider channel. The power manager is connected to a double inductor through two consumer channels. The dotted lines show the metamodel concepts that are related to the model. The bottom part shows the encoding of the model. This encoding will be used to define model fragments on the parent model, which is explained in the next subsection.

2.2 | Search strategy: an evolutionary algorithm for MFLP

To develop an evolutionary algorithm, the following elements must be defined: 1) an encoding of the problem that can be used to represent the individuals; 2) a fitness function that can be used to evaluate how good each individual is as a solution to the problem; and 3) a set of genetic operators that can be applied to modify and evolve the population of individuals.

Figure 2 shows an overview of the evolutionary algorithm used. The left part shows the inputs for the approach: a MFLP description and a set of product models. The center shows a simplified representation of the main steps. The 'Initialize Population' step calculates an initial population of model fragments from the input set of product models. The 'Genetic Operations' step produces the new generation of model fragments. Finally, the 'Fitness' step assigns values that assess how good each model

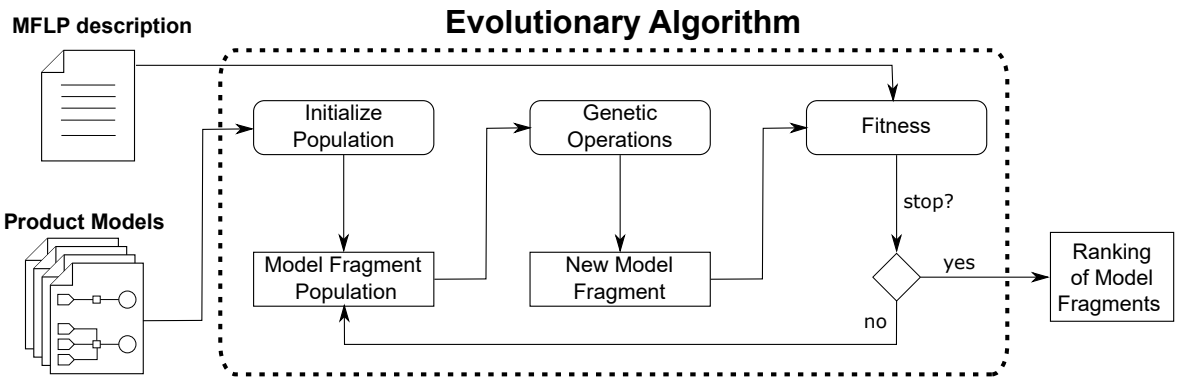


FIGURE 2 Overview of the search strategy.

fragment is based on the description. As output, the approach provides a list of model fragments that might be relevant as a solution for the location problem.

2.2.1 | Encoding

In this case, the candidates are model fragments. To represent a candidate solution (individual), we use a binary string. Each position of the binary string represents a model element (concept, properties, or relationships) related to a parent model. Thus, a candidate solution is defined as a set of model elements that is associated with a parent model. The size of the solution represents the number of model elements in the string. This representation defines model fragments, indicating the presence or absence of each of the model elements of the parent.

Model Fragment 1 in Figure 1 shows a model fragment that was defined in the Parent Model. All of the model fragments are defined concerning a parent model. When the position is equal to 0, it indicates that the element is not part of the model fragment. However, when the position is equal to 1, it indicates that the element is part of the model fragment. Thus, we can indicate the model elements from the model parent that compose the model fragment. For instance, Model Fragment 1 is a model fragment that contains the power manager, the two consumer channels, and the two inductors of the parent model.

2.2.2 | Fitness function

The search strategy is guided using a fitness function that is based on textual comparisons[9]. In this case, the algorithm assesses the relevance of each model fragment in relation to the description provided by the software engineers of our industrial partners (see the upper-right part of Figure 1).

The description uses natural language. To assess the relevance of each model fragment to the description, we apply a technique that is based on information retrieval. First, the text from the textual description and the models is homogenized through the use of natural language processing techniques[12]. Then, we apply Latent Semantic Indexing (LSI)[13] to analyze the relationship between the description and the solution candidates generated. The result is a ranking of model fragments that are ordered based on the similitude with the description.

2.2.3 | Genetic operators

The genetic manipulation of the individuals is performed using four different operators. These operators were defined to work on model fragments[14,9]:

- **Parent Selection:** This operator is responsible for selecting the parent individuals that will serve as the base for the new offspring. In this approach, we use a generic roulette wheel selection, which assigns a probability of being selected that is proportional to each individual's fitness value. All individuals can be selected as parents, but the higher the fitness value, the higher the probability of being selected as a parent for the next generation.

- **Crossover:** The crossover operator aims to combine the genetic material from the parents into new individuals. In this approach, we use a multiple-point crossover operator that is based on a mask [14] that will combine two model fragments into two new individuals. The mask determines how the combination is done. For each element of the model fragments, it indicates if the offspring should inherit from one parent or the other (including/excluding the element depending on whether or not the element is present in the parent). As a result, two individuals are generated, one by applying the mask directly and another one by applying the inverse of the mask. The crossover operator is not always applied to the new offspring; the crossover rate parameter (p_c) determines whether or not it is applied.

- **Mutation:** The mutation operator aims to imitate the spontaneous mutations that occur in nature. The mutation can turn into an advantage or a disadvantage for the survival of the individual. In this approach, we use an evenly distributed mutation across the genes of the individual. The mutation operator can perform two kinds of modifications: the addition of elements to the fragment (by inverting a 0 for a 1 in the binary string), or the removal of elements from the model fragment (by inverting a 1 for a 0 in the binary string). Again, the mutation is not always applied to the new offspring; the mutation rate parameter (p_m) determines whether or not it is applied.

- **Replacement:** The replacement operator aims to modify the current population by combining it with the new offspring generated with previous operators. In this approach, we replace the less fit part of the current population with new offspring. Two parameters will determine the outcome of the operator: the population size (μ), which will be kept constant throughout the entire execution of the search, and the replacement factor (λ), which determines the number of individuals that are replaced in each generation.

2.3 | Measures that define model fragment location problems (MD-MFLP)

Some research works [10,6] use a set of five measures that define and characterize model fragment location problems (MFLP). Two of the five measures characterize the model parent where the MFLP is going to be located (search space), and the other three measures characterize the model fragment that realizes the MFLP (solution). These measures are the followings:

Search Space Size (SS-Size) measures the number of elements of the model where the MFLP is located. The elements are concepts, properties, or relationships that are present in the parent model. The SS-Size value of the model presented in Figure 1 is 14, which matches the number of bits in the encoding.

Search Space Volume (SS-Volume) measures the number of models that compose the search space. The location of a MFLP may be not limited to a single model and sometimes has to be performed in more than one model. In the example from Figure 1, the SS-Volume value is 1 because the search is performed in one model. If it is necessary to locate the same MFLP in more than one model, one model fragment is obtained for each model used in the location.

Model Fragment Density (MF-Density) measures the percentage of model elements that realize a solution, in other words, the ratio of model elements in the model fragment and the model elements of the parent model. In the example from Figure 1, the MF-Density value is 0.64 (9 model elements in the model fragment divided by 14 model elements in the parent model).

Model Fragment Multiplicity (MF-Multiplicity) measures the number of times the solution appears in the search space. In the example from Figure 1, the MF-Multiplicity value is 1 because the MFLP only appears once.

Model Fragment Dispersion (MF-Dispersion) measures the ratio of connected elements in the solution. It is computed as the ratio between the number of groups and the number of model elements of the model fragment. In the example from Figure 1, the MF-Dispersion value is 0.07 (1 group in the model fragment divided by 14 model elements in the model fragment). This value can range from 0 to 1. Values around 0 indicate a strong connection among the model fragment elements, while values around 1 indicate a strong dispersion among the model fragment elements.

In this work, we characterize the case studies and the results obtained with these specific measures that define model fragment location problems. In addition, we focus on the search parameters from the genetic operators that can be tuned to boost the performance of the approach.

Quantitative parameters	Set of values
Population size (μ)	50, 100, 150, 200, 250
Replacements (λ)	20%, 30%, 40%, 50%, 60%
Mutation rate (p_m)	0.2, 0.4, 0.6, 0.8, 1.0
Crossover rate (p_c)	0.2, 0.4, 0.6, 0.8, 1.0

TABLE 1 Representation of quantitative parameters representation.

2.4 | Search parameters of the evolutionary algorithm for model fragment location

Evolutionary algorithm researchers acknowledge that good search parameter values are essential for good evolutionary algorithm performance [15]. In addition, the literature distinguishes between qualitative parameters (finite domain with no sensible distance or ordering) and quantitative parameters (infinite domain with structure or order) [15].

For example, in our approach, we define our own qualitative parameter for the crossover operator parameter, which describes how the crossover will be done (the operator itself). However, since the crossover rate expresses the probability of the crossover operator occurring and can take any value between 0 and 1, it is considered a quantitative parameter. For both types of parameters, the elements of the parameter's domain are called parameter values and we instantiate a parameter by allocating a value to it. In this case study, the qualitative parameters (operators indicated in previous sections) will be fixed from the beginning; we are going to evaluate the search approach using different parameter values for the quantitative parameters.

Table 1 shows the most common quantitative search parameters that are used in current practice that will be used to drive our approach. The population size (μ) parameter describes the number of model fragments that are maintained as candidate solutions in each generation. The replacement (λ) parameter indicates the percentage of the population that is going to be discarded in the next iteration. Therefore, it also determines the number of new individuals that will be generated through the application of the genetic operators. The mutation rate (p_m) and the crossover rate (p_c) define the probability of applying the mutation operator or the crossover operator in each case.

A set of values for each one of the parameters is known as a parameter vector. In our case, the parameter vector $p = \{\mu, \lambda, p_m, p_c\}$ can take the values defined in Table 1 for each one of the search parameters. The values considered for the experimentation are considered to be common values in the literature [16,17]. For each search parameter, we included both high and low values to properly explore the alternatives. During the evaluation, 625 different parameter vectors are used, so all of the combinations of values are explored. (e.g., $p1 = \{100, 20\%, 0.2, 0.8\}$, $p2 = \{150, 10\%, 0.4, 0.6\}$).

There are two atomic performance measures for Evolutionary Algorithms. The first one measures the solution quality (already evaluated in our previous work [18,14]), and the second one measures the algorithm speed or search effort (evaluated in [8]). In this work, we try to maximize the quality of the solution obtained by the approach by varying the parameter vectors. The approach is executed a fixed amount of time using one of the parameter vectors. Then, the quality of solutions achieved is compared to determine which parameter vector provides the best quality.

3 | EXPERIMENTAL SETUP

This section presents the process followed to evaluate the impact of the different parameter values taking into account the measures that define model fragment location problems.

RQ1 Is there any impact on performance regarding the search parameter vector for model fragment location problems?

RQ2 Is there any impact on performance regarding the measures that define model fragment location problems?

RQ3 Are there any relations between the measures that define the model fragment location problems and the search parameter vector used that affect the performance?

To evaluate the impact of the different parameter values on the results, we execute the approach using a different set of parameter values each time to solve 1,895 MFLP from two of our industrial partners: BSH, the leading manufacturer of home appliances in Europe; and CAF, an international provider of railway solutions worldwide. In addition, we perform a statistical analysis to ensure the validity of the results.

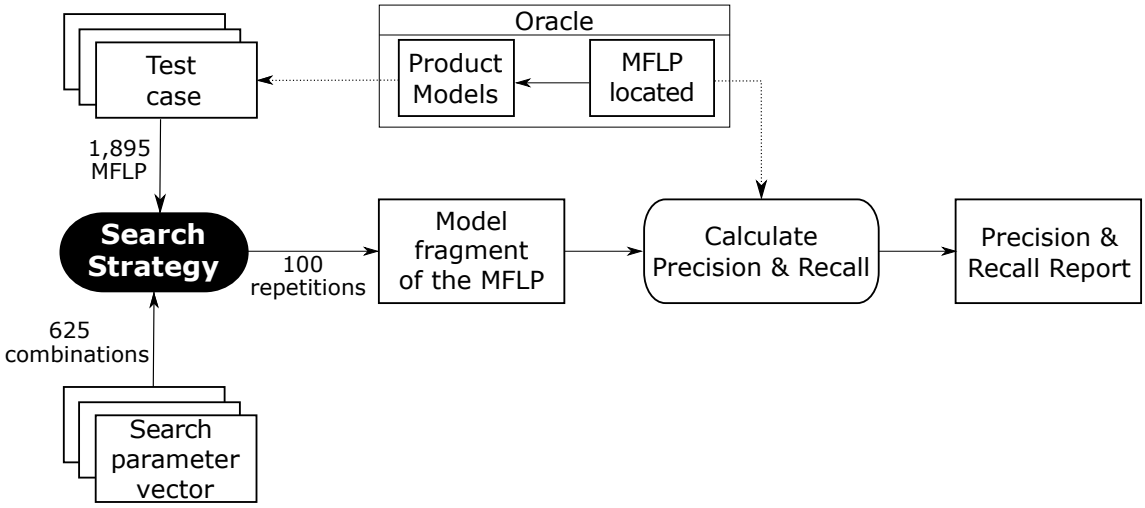


FIGURE 3 Overview of the evaluation with the oracle.

We use the product models from our industrial partners as an oracle to evaluate the results. In other words, we make use of a set of product models whose MFLP realizations are known beforehand and that are considered to be the ground truth, thus allowing us to compare the results provided by our approach with the oracles.

3.1 | Setup of the case studies

Figure 3 shows an overview of the process that we have followed in the evaluation. The top-right part presents the oracle (set of product models with the MFLP located and formalized).

First, we construct a test case for each MFLP that is present in the oracle. In addition, we generate the 625 different parameter vectors with values from Table 1 to configure the algorithm (left part of Figure 3). We run each test case for each one of the possible parameter vectors for a fixed time. The allocated time is 10 s (a prior test showed that the search converges in less than the allocated time). This results in a model fragment for each of the test cases for each parameter vector. Finally, the solutions are compared with the model fragments from the oracle (considered the ground truth) in order to obtain the precision, recall, and F-measure values. The operation is repeated 100 times for each combination of the parameter vector and test case to reduce the stochastic component that algorithms of this type have. Finally, the data is aggregated into a report containing the results of the executions (shown in Section 4.1).

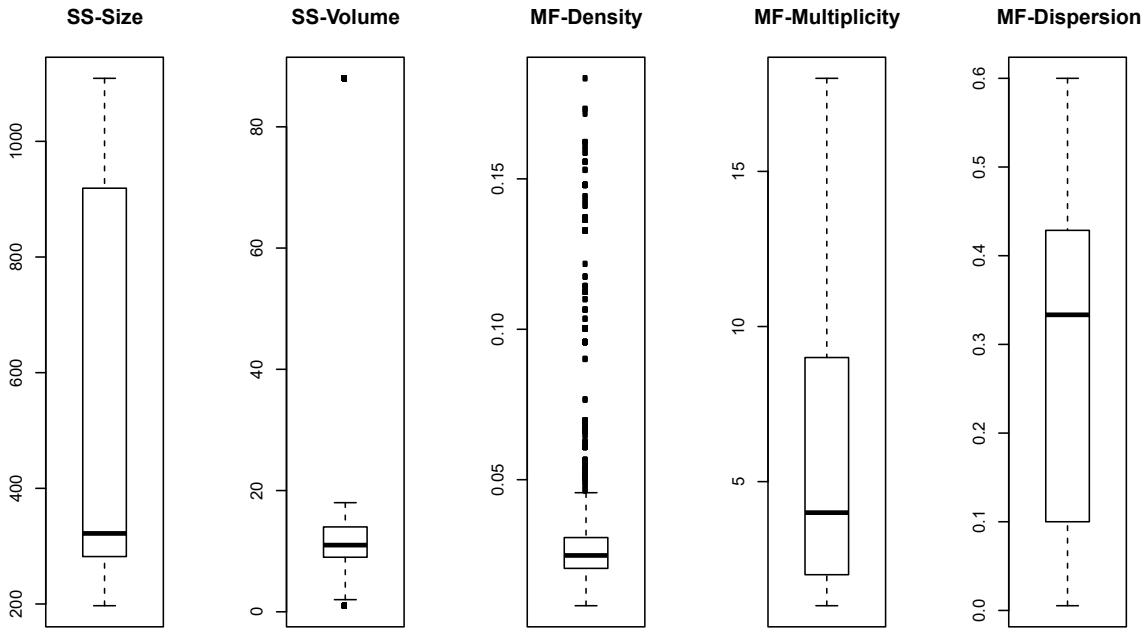
The Model Fragment Location Problems (MFLPs) used in this evaluation were obtained from two of our industrial partners, BSH and CAF:

3.1.1 | BSH

BSH is one of the leadering companies in the home appliances sector. We collaborated with the induction division, which has been creating the firmware of Induction Hobs (IHs) for brands like Bosch and Siemens for the last fifteen years. The latest firmware produced includes full cooking surfaces, where heating areas are dynamically created and activated or deactivated depending on the characteristics (size, position, material, etc.) of the cookware placed on top.

There has also been an increase in the feedback that the hob provides to the cook, such as temperatures of the food being cooked in the cookware or even real-time values of the actual consumption of the Induction Hob. In this evaluation, we use 608 MFLPs extracted from the products that they develop. The oracle is composed of the description of each MFLP, the models of the products, and the model fragment that corresponds to each of the MFLP[1].

[1]The following video shows the IH models and model fragments used by BSH:www.youtube.com/watch?v=nS2sybEv6j0



Measure	Search Space (SS)		Model Fragment (MF)		
	Size	Volume	Density	Multiplicity	Dispersion
Min	197.0	1.00	0.0081	1.000	0.0053
1st Q	282.0	9.00	0.0205	2.000	0.1000
Median	322.0	11.00	0.0248	4.000	0.3333
Mean	515.7	14.48	0.0316	5.993	0.2784
3rd Q	919.0	14.00	0.0308	9.000	0.4286
Max	1,109.0	88.00	0.1834	18.000	0.6000
Sd	327.91	16.63	0.0271	4.569	0.1709

FIGURE 4 Box plots with the value for each MD-MFLP obtained from the 1,895 MFLPs and report of the search problems used in the evaluation.

3.1.2 | CAF

We also use the models from CAF in our evaluation. CAF is a constructor of railway solutions. They produce trains in many different forms (regular trains, subway, light rail, monorail, etc.) that are distributed worldwide. A train includes different pieces of specific equipment to carry out specific tasks for the train. These are located in vehicles and cabins and are usually designed and manufactured by different providers.

The DSL used by CAF can be used to describe the interaction between the different pieces of equipment of the train. Moreover, the DSL allows specifying non-functional aspects that are related to regulations, such as the different levels of redundancy present in the system or the quality of signals from the equipment. This results in a DSL that is composed of around 1,000 different elements.

In this evaluation, we use 1,287 MFLPs extracted from the products that they develop. The oracle is composed of the description of each MFLP, the models of the products, and the model fragment that corresponds to each of the MFLPs [2].

[2]The following video shows the train models and model fragments used by CAF: www.youtube.com/watch?v=Ypcl2evEQB8

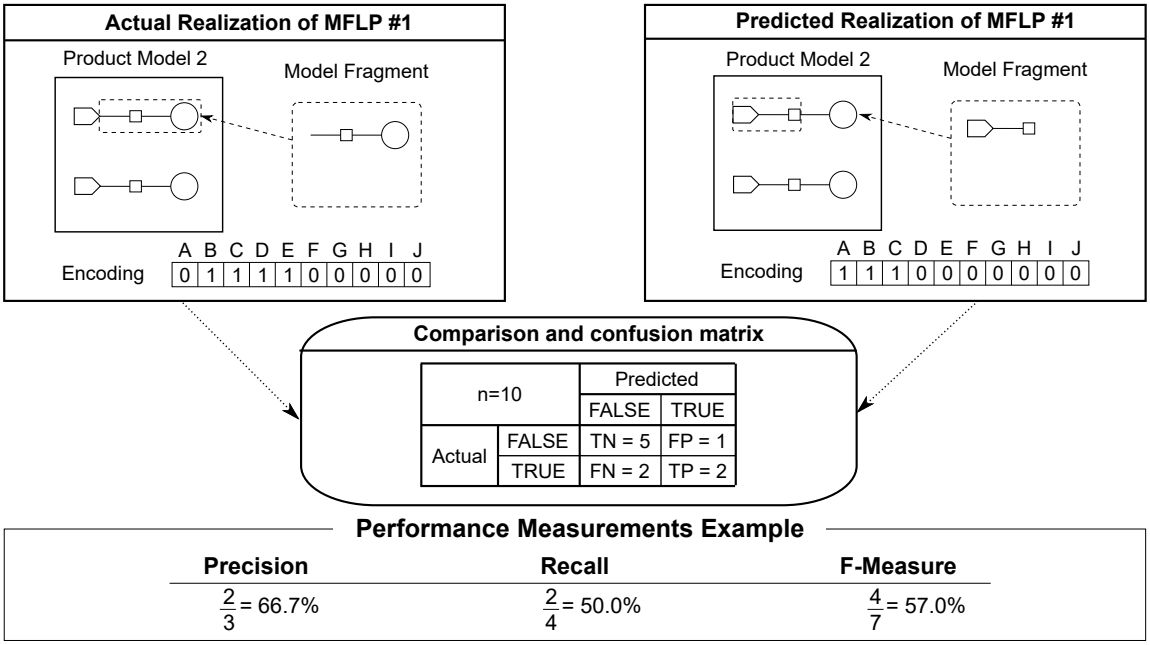


FIGURE 5 Example of a confusion matrix for two model fragments.

Therefore, in this evaluation, we have 1,895 different MFLPs provided by our industrial partners. The use of two different domains with a wide variety of MFLPs and their casuistry has led to an improvement in the generalizability of our assessment.

We classified each of the MFLPs by means of the Measures that define Model Fragment Location Problems (MD-MFLP) (Section 2.3). For each MD-MFLP, we define two groups: HIGH and LOW. To do this, we use a median-based discretization by splitting each MD-MFLP by the value of the sample median[19,20]. Figure 4 show the values for each of the five MD-MFLPs of the MFLPs used in this evaluation. Values above the median are considered to be HIGH, while values below the median are considered to be LOW. For instance, the median of the MF-Multiplicity of all of the MFLPs is 4; all MFLPs whose MF-Multiplicity value are above 4 are defined as HIGH for MF-Multiplicity; in contrast, all MFLPs whose MF-Multiplicity value are below 4 are defined as LOW for MF-Multiplicity.

3.2 | Performance measurements

Once the results from applying the approach to the test cases are obtained, we proceed to compare them with the oracle and measure them in terms of some software quality properties. Figure 5 shows an example of a model fragment from the oracle (left), a model fragment candidate obtained from the application of the approach (right), and the confusion matrix[21] used to compare both (middle).

A confusion matrix is a table that is often used to describe the performance of a classification model (our approach under evaluation) on a set of test data (the resulting model fragments) for which the true values are known (from the oracle). In this case, each MFLP realization returned by the approach is a model fragment that is composed of a subset of the model elements that are part of the product model (where the MFLP is being located). Since the granularity will be at the level of model elements, the presence or absence of each model element will be considered as a classification. Therefore, our confusion matrices will distinguish between two values: TRUE (presence) and FALSE (absence).

Figure 5 shows an example of the comparison process performed to compare a result from one of the evaluated approaches with the ground truth from the oracle and the resulting confusion matrix. The left part shows the actual realization of MFLP #1 (obtained from the oracle and considered the ground truth), while the right part shows the predicted realization of MFLP #1 output by the approach. The confusion matrix arranges the results of the comparison into four categories:

True positive (TP): A model element present in the predicted realization that is also present in the actual realization (e.g., model element B is a TP).

True Negative (TN): A model element not present in the predicted realization that is also not present in the actual realization (e.g., model element H is a TN).

False Positive (FP): A model element present in the predicted realization that is not present in the actual realization (e.g., model element A is an FP).

False Negative (FN): A model element not present in the predicted realization that is present in the actual realization (e.g., model element D is an FN).

The confusion matrix holds the results of the comparison between the predicted results and the actual results; it is just a specific table layout to help the visualization of the performance of a classifier. However, to evaluate the performance of the approach, it is necessary to derive some measurements from the values of the confusion matrix, which in this case are the three measurements are precision, recall, and F-measure.

Precision measures the number of elements from the prediction (the result of the approach) that are correct according to the ground truth (the oracle).

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

Recall measures the number of elements of the ground truth (the oracle) that are correctly retrieved by the prediction (the result of the approach).

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

F-measure combines both recall and precision as the harmonic mean of precision and recall.

$$F - measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3}$$

Precision and recall values can range between 0% to 100%. Following up with the example of the confusion matrix in Figure 5, we can calculate the precision, recall, and F-measure for the model fragment (see Figure 5). The model fragment has a measurement of 66.7% in precision (two out of the three elements included in the candidate model are present in the model fragment from the oracle) and 50% in recall (2 out of the 4 elements that are present in the oracle are also present in the model fragment). This results in a combined F-measure of 57%.

Details about the implementation of the search strategy approach can be seen in[8,14]. We performed the execution of the approach using an array of computers with 8 core processors, clock speeds of 4GHz, and 16 GB of RAM. All of them were running Windows 10 Pro N 64 bits as the hosting Operative System and the Java(TM) SE Runtime Environment (build 1.8.0_73-b02).

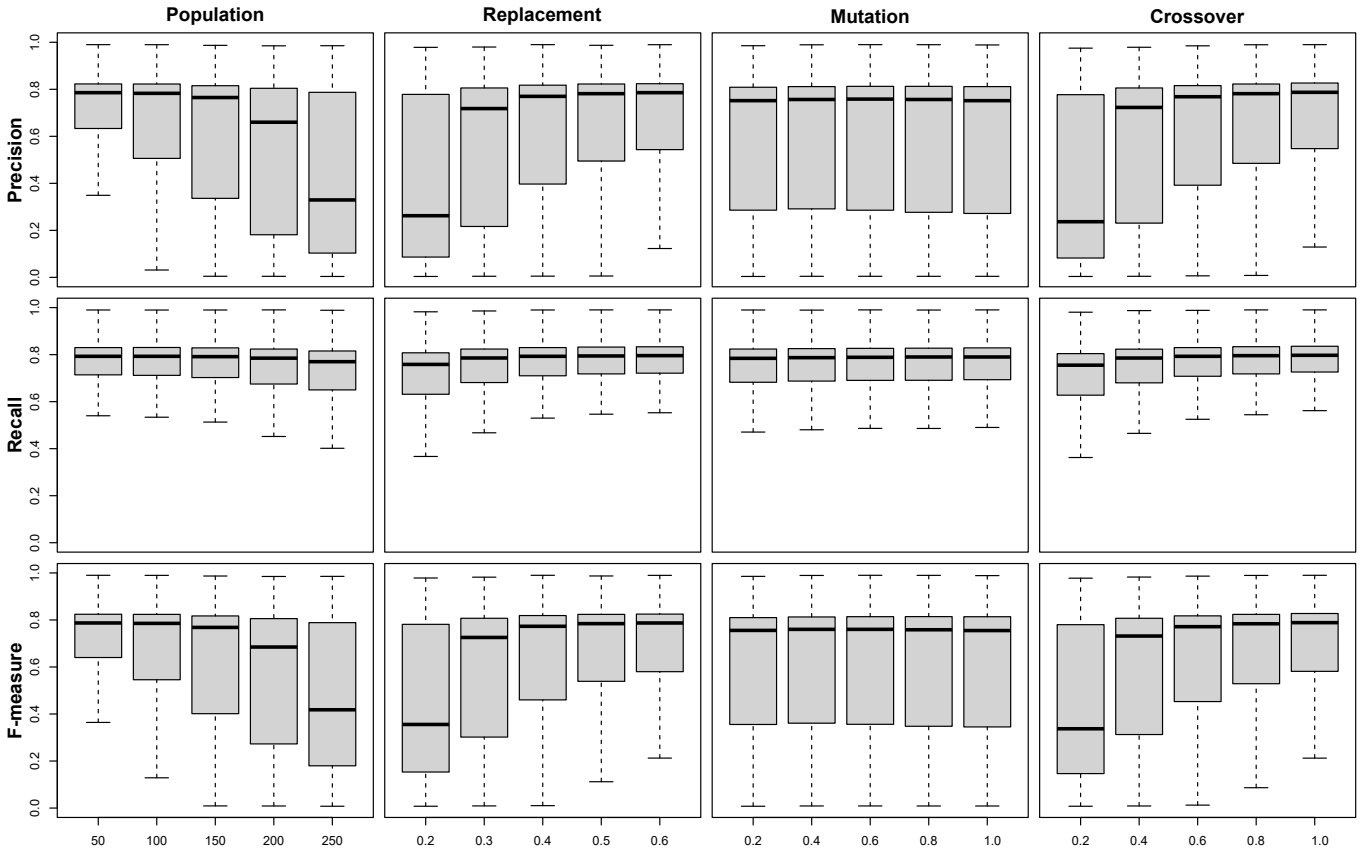
Since the software models of the case studies are currently operating or will be released in the near future, this information is limited by confidentiality agreements with our industrial partners. Nevertheless, for purposes of replicability, the CSV files with the results of this evaluation and that were used as input in the statistical analysis are published online at: https://svit.usj.es/SPE_Roca_data/.

4 | RESULTS

This section presents the results obtained in the evaluation and the statistical analysis that answers each research question, the findings, the discussion of the results, and the threats to validity.

4.1 | Evaluation results

This section is divided into three parts. The first part shows the performance of the four search parameters at each of their five possible values. The second part shows the performance of the MD-MFLP measures for the two levels established (LOW and HIGH). The third part shows the performance taking into account both the search parameters and the measures.



Parameters		Precision ± (σ)	Recall ± (σ)	F-measure ± (σ)
Population size μ (50-250)	$\mu_1 = 50$	68.45 ± 21.85	73.26 ± 16.10	69.67 ± 20.14
	$\mu_2 = 100$	65.49 ± 24.98	73.23 ± 16.04	67.14 ± 22.81
	$\mu_3 = 150$	59.86 ± 28.50	72.98 ± 15.90	62.35 ± 25.81
	$\mu_4 = 200$	51.90 ± 31.26	72.33 ± 15.75	55.40 ± 28.37
	$\mu_5 = 250$	42.02 ± 32.06	71.20 ± 15.62	46.58 ± 29.43
Replacement percentage λ (20-60)	$\lambda_1 = 20\%$	39.32 ± 32.44	70.03 ± 15.75	43.79 ± 29.99
	$\lambda_2 = 30\%$	54.14 ± 30.47	72.32 ± 15.77	57.37 ± 27.58
	$\lambda_3 = 40\%$	62.03 ± 26.92	73.28 ± 15.85	64.32 ± 24.28
	$\lambda_4 = 50\%$	65.37 ± 24.69	73.62 ± 15.90	67.20 ± 22.32
	$\lambda_5 = 60\%$	66.84 ± 23.51	73.76 ± 15.93	68 46 ± 21 32
Mutation rate p_m (0-1)	$p_{m1} = 0.2$	57.65 ± 29.48	72.21 ± 15.97	60.25 ± 26.86
	$p_{m2} = 0.4$	58.00 ± 29.42	72.46 ± 15.96	60.57 ± 26.76
	$p_{m3} = 0.6$	57.82 ± 29.55	72.61 ± 15.99	60.45 ± 26.84
	$p_{m4} = 0.8$	57.34 ± 29.72	72.72 ± 15.95	60.09 ± 26.97
	$p_{m5} = 1.0$	56.90 ± 29.79	73.01 ± 15.62	59 78 ± 26 98
Crossover rate p_c (0-1)	$p_{c1} = 0.2$	38.77 ± 32.80	69.58 ± 15.77	43.26 ± 30.41
	$p_{c2} = 0.4$	54.63 ± 30.15	72.09 ± 15.94	57.84 ± 27.28
	$p_{c3} = 0.6$	61.78 ± 26.75	73.21 ± 15.89	64.12 ± 24.10
	$p_{c4} = 0.8$	65.22 ± 24.63	73.83 ± 15.81	67.07 ± 22.23
	$p_{c5} = 1.0$	67.32 ± 23.21	74.30 ± 15.65	68.85 ± 20.98

FIGURE 6 Box plots, mean values, and standard deviation of the precision, recall, and F-Measure values obtained by each search parameter.

Ranking	Parameters				Performance		
	μ	λ	p_m	p_c	Precision	Recall	F-measure
1	150	50%	1.0	1.0	73.08	75.45	73.64
2	200	50%	1.0	1.0	73.01	75.51	73.62
3	150	60%	1.0	1.0	72.98	75.34	73.53
4	150	40%	1.0	1.0	72.92	75.38	73.53
5	250	60%	1.0	1.0	72.70	75.80	73.53
				...			
621	250	20%	0.2	0.2	7.12	62.53	12.10
622	250	20%	0.4	0.2	6.79	62.24	11.74
623	250	20%	0.6	0.2	6.63	61.91	11.52
624	250	20%	0.8	0.2	6.59	62.33	11.47
625	250	20%	1.0	0.2	6.48	62.09	11.31

TABLE 2 Search parameter vector performance ranking ordered by F-measure.

4.1.1 | Performance by search parameter value

The bottom part of Figure 6 shows the mean values and standard deviations that were calculated for each search parameter value. We also show the F-measure value that combines precision and recall into a single performance indicator.

For the population size, the $\mu_1 = 50$ value achieved the best results reaching mean values of 68.45% in precision, 73.26% in recall, and 69.67% in F-measure. However, all of the values provided similar results in terms of recall with differences below 5%.

For the replacement percentage, the $\lambda_5 = 60\%$ value achieved the best results reaching mean values of 66.84% in precision, 73.76% in recall, and 68.46% in F-measure. Again, the differences in terms of recall when using each of the parameter values were small (below 5%).

For the mutation rate, the $p_{m2} = 0.4$ value achieved the best results in terms of precision with a mean of 58.00%, while the $p_{m5} = 1.0$ value achieved the best results in terms of recall with a mean of 73.01%. However, the differences in performance when using each of the parameter values were small (below 5%).

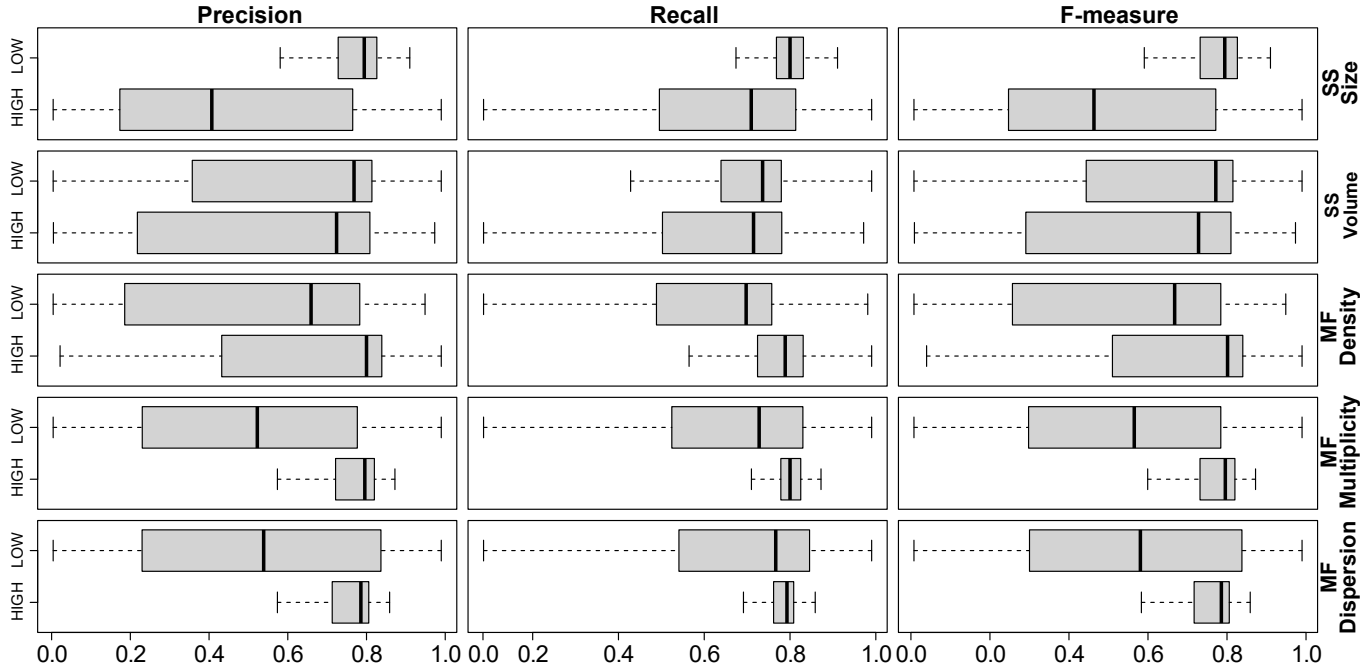
For the crossover rate, the $p_{c5} = 1.0$ value achieved the best results reaching mean values of 67.32% in precision, 74.30% in recall, and 68.85% in F-measure. Again, the differences in terms of recall when using each of the parameter values were small (below 5%).

Table 2 shows best and worst performance means by each search parameter vector ordered by F-measure. The top five search parameter vectors have the values of mutation rate and the crossover rate in common, (i.e., both are equal to 1.0). The bottom five search parameter vectors have the values of population size ($\mu_5 = 250$), replacement percentage $\lambda_1 = 20\%$, and crossover rate ($p_{c1} = 0.2$) in common. However, the mutation rate takes all possible values.

4.1.2 | Performance by MD-MFLP

The bottom part of Figure 7 shows the mean values and standard deviation of the precision, recall, and F-measure values obtained by each MD-MFLP. In the measures related to the search space, LOW values obtained better values in performance, while in the measure related to the model fragment, HIGH values obtained better values.

For the SS-Size, the differences between HIGH and LOW values were above 22 points, obtaining a mean of 70.55% in F-measure for LOW values. For the SS-Volume, the differences between HIGH and LOW values were above 6 points, obtaining a mean of 63.17% in F-measure for LOW values. For the MF-Density, the differences between HIGH and LOW values were above 14 points, obtaining a mean of 67.41% in F-measure for HIGH values. For the MF-Multiplicity, the differences between HIGH and LOW values were above 13 points, obtaining a mean of 67.6% in F-measure for HIGH values. Finally, for the MF-Dispersion, the differences between HIGH and LOW values were above 11 points, obtaining a mean of 66.91% in F-measure for HIGH values.



MD-MFLP	Value	Precision ± (σ)	Recall ± (σ)	F-measure ± (σ)
SS-Size	LOW	68.79 ± 24.73	79.16 ± 5.85	70.55 ± 21.60
	HIGH	44.46 ± 29.41	64.97 ± 19.97	48.22 ± 27.43
SS-Volume	LOW	60.40 ± 28.35	74.81 ± 13.25	63.17 ± 25.42
	HIGH	54.05 ± 30.68	69.90 ± 18.28	56.63 ± 28.16
MF-Density	LOW	50.59 ± 30.29	67.50 ± 17.62	53.08 ± 28.06
	HIGH	64.53 ± 27.13	77.74 ± 11.93	67.41 ± 23.55
MF-Multiplicity	LOW	50.51 ± 29.35	66.90 ± 19.44	53.70 ± 27.15
	HIGH	65.55 ± 27.78	79.10 ± 5.58	67.66 ± 24.55
MF-Dispersion	LOW	51.93 ± 30.28	68.77 ± 19.68	55.22 ± 27.91
	HIGH	65.03 ± 26.89	77.72 ± 5.28	66.91 ± 23.86

FIGURE 7 Box plots, mean values, and standard deviation of the Precision, Recall, and F-Measure values obtained by each MD-MFLP. Each measure is reported as a two-value factor (HIGH and LOW).

Table 3 shows the ranking of performance means by each MD-MFLP combination ordered by F-measure. Some of the 32 scenarios are missing due to the lack of test cases with those combinations of MD-MFLPs. The top five best results were achieved when the SS-Size was equal to LOW and MF-Density was equal to HIGH. The bottom five results coincide in SS-Size with the opposite value (equal to HIGH) and the MF-Multiplicity value equal to LOW.

4.1.3 | Performance by search parameter value and MD-MFLP

Table 4 presents the mean values of precision, recall, and F-measure achieved by each search parameter value and each MD-MFLP. Similarly, Figure 8 shows the graphs obtained from those values. The results show that the performance by MD-MFLP remained the same as in the previous subsection. In the measures related to the search space, LOW values obtained better values in performance, while, in the measure related to the model fragment HIGH values obtain better values.

Ranking	Search Space		Model Fragment			Performance		
	Size	Volume	Density	Multiplicity	Dispersion	Precision	Recall	F-measure
1	LOW	LOW	HIGH	LOW	LOW	73.19	84.54	75.55
2	LOW	LOW	HIGH	LOW	HIGH	73.15	79.48	74.35
3	LOW	HIGH	HIGH	LOW	LOW	72.24	82.70	74.26
4	LOW	HIGH	HIGH	HIGH	LOW	71.65	82.68	73.74
5	LOW	HIGH	HIGH	LOW	HIGH	71.83	79.53	73.23
				...				
20	HIGH	LOW	HIGH	LOW	LOW	48.76	70.97	55.28
21	HIGH	LOW	LOW	LOW	HIGH	51.15	71.64	53.34
22	HIGH	HIGH	HIGH	LOW	LOW	49.35	64.73	52.54
23	HIGH	LOW	LOW	LOW	LOW	38.80	54.87	41.47
24	HIGH	HIGH	LOW	LOW	LOW	34.05	55.63	37.40

TABLE 3 MD-MFLP performance ranking ordered by F-measure.

The same occurred with the population size, replacement percentage, and crossover rate search parameters. The values that obtained the best performance for them are the ones described in section 4.1.1. However, the best performing values for mutation rate were not those described in section 4.1.1 for all cases. The $p_{m1} = 0.2$ achieved the best results in terms of precision for SS-Size, MF-Multiplicity, and MF-Dispersion.

Table 5 shows best and worst performance means for each combination of search parameter vector and MD-MFLP ordered by the F-measure value. The top five values of performance correspond to the same scenario. Again, in the scenario, shares that the SS-Size value was LOW and the MF-Density value was HIGH. With regard to the search parameter values, the top five values of performance had the same crossover rate value ($p_{c5} = 1.0$). Similarly, the bottom five values correspond to the same scenario. In contrast to the previous scenario, the SS-Size value was HIGH and the MF-Density value was LOW. Moreover, the bottom five values had the same crossover rate value ($p_{c1} = 0.2$), the same population size value ($\mu_5 = 250$), and the same replacement percentage value ($\lambda_1 = 20\%$). Note that the crossover value was the complete opposite of the previous one ($p_{c1} = 0.2$ bottom values vs. $p_{c5} = 1.0$ top values).

4.2 | Statistical analysis

The data resulting from the empirical analysis has been analyzed using statistical methods following the guidelines in[22]. The goals of the analysis are: (1) to provide formal and quantitative evidence (statistical significance) regarding whether or not the different search parameter vectors and the different metrics have an impact on the performance; and (2) to show that those differences are significant in practice (effect size).

To enable statistical analysis, the algorithm must be run a large enough number of times (in an independent way) to collect information on the probability distribution for each search parameter vector. Then, a statistical test is run to assess whether there is enough empirical evidence to claim (with a high level of confidence) that there is a difference in the performance.

4.2.1 | Statistical significance

To analyze the statistical significance of the different search parameter vectors and the different measures, we used the ANOVA test. To apply this test, it is important to comply with normality and homogeneity of variance. The box plots depicted in the previous sections show that the data does not follow a normal distribution. However, the Central Limit Theorem states that the sampling distribution of means is normally distributed for large enough samples[23]. In our case, normality of sampling distributions is ensured by having sufficiently large sample sizes.

The result of the ANOVA test showed that there were statistically significant differences between groups because, $p - values$ obtained by precision, recall and F-measure values were lower than 0.05 for all of the search parameter values and MD-MFLP.

As we detected significant differences between the groups, we tested which groups were significantly different. We performed an additional post hoc analysis, which consisted of a pair-wise comparison among the results for each search parameter and for

			Population size (μ)				Replacement percentage (λ)					Mutaion rate (p_m)					Crossover rate (p_c)				
			50	100	150	200	250	20%	30%	40%	50%	60%	0.2	0.4	0.6	0.8	1.0	0.2	0.4	0.6	0.8	1.0
SS-Size	LOW	P	79.54	77.64	72.49	62.96	51.32	49.04	65.96	74.01	76.87	78.06	69.94	69.66	69.06	68.21	67.08	53.05	67.29	72.45	74.86	76.29
		R	79.85	79.93	79.65	78.85	77.53	76.84	79.04	79.80	80.02	80.12	78.95	79.12	79.21	79.27	79.28	77.19	78.99	79.62	79.91	80.10
		F	79.65	78.10	73.83	65.72	55.46	53.10	68.32	75.19	77.58	78.57	71.48	71.24	70.76	70.08	69.21	56.86	69.34	73.74	75.79	77.02
	HIGH	P	55.56	51.36	45.16	39.03	31.20	28.02	40.39	48.09	52.01	53.80	43.36	44.44	44.75	44.71	45.05	22.16	39.90	49.36	54.00	56.88
		R	65.60	65.44	65.23	64.74	63.84	62.11	64.50	65.70	66.18	66.36	64.38	64.73	64.93	65.09	65.72	60.73	64.06	65.75	66.76	67.56
		F	58.06	54.38	48.99	43.40	36.25	32.96	44.64	51.67	55.13	56.70	47.19	48.16	48.47	48.47	48.81	27.44	44.47	52.92	56.92	59.34
SS-Volume	LOW	P	71.93	69.00	63.01	54.33	43.72	41.61	56.72	65.00	68.52	70.14	60.73	60.99	60.70	60.12	59.46	42.10	57.43	64.43	67.93	70.10
		R	75.67	75.61	75.23	74.39	73.13	72.16	74.49	75.50	75.86	76.03	74.48	74.73	74.83	74.91	75.09	72.02	74.35	75.38	75.95	76.33
		F	73.08	70.62	65.56	58.01	48.59	46.33	60.11	67.35	70.35	71.71	63.41	63.64	63.40	62.95	62.45	46.85	60.79	66.84	69.78	71.59
	HIGH	P	64.19	61.18	56.00	48.92	39.93	36.52	50.99	58.39	61.52	62.81	53.88	54.34	54.30	53.95	53.76	34.69	51.19	58.53	61.91	63.91
		R	70.31	70.32	70.23	69.81	68.84	67.43	69.66	70.56	70.89	70.97	69.44	69.70	69.89	70.03	70.46	66.59	69.32	70.54	71.25	71.82
		F	65.50	62.87	58.43	52.21	44.12	40.68	54.02	60.61	63.35	64.48	56.38	56.82	56.84	56.58	56.50	38.87	54.23	60.79	63.75	65.49
MF-Density	LOW	P	62.24	58.85	52.60	44.46	34.77	32.38	46.95	55.00	58.53	60.08	50.62	50.99	50.82	50.35	50.15	30.65	47.72	55.28	58.62	60.66
		R	68.11	68.08	67.83	67.23	66.23	65.12	67.25	68.13	68.43	68.56	67.11	67.35	67.46	67.56	68.00	64.57	67.04	68.10	68.64	69.12
		F	63.53	60.49	55.01	47.66	38.74	36.08	49.89	57.26	60.40	61.79	53.02	53.40	53.29	52.92	52.79	34.39	50.64	57.53	60.52	62.34
	HIGH	P	74.69	72.16	67.15	59.37	49.30	46.30	61.38	69.09	72.26	73.65	64.72	65.05	64.86	64.37	63.68	46.94	61.57	68.30	71.85	74.01
		R	78.44	78.42	78.17	77.46	76.20	74.97	77.41	78.46	78.85	78.99	77.34	77.61	77.79	77.90	78.05	74.62	77.16	78.34	79.05	79.52
		F	75.84	73.82	69.74	63.19	54.46	51.54	64.90	71.41	74.04	75.16	67.51	67.79	67.66	67.29	66.79	52.18	65.08	70.74	73.65	75.39
MF-Multiplicity	LOW	P	58.90	56.01	51.94	46.47	39.23	34.92	47.76	54.36	57.14	58.37	49.68	50.56	50.76	50.65	50.89	31.14	47.18	54.83	58.57	60.82
		R	67.22	67.17	67.13	66.84	66.12	64.58	66.61	67.51	67.83	67.95	66.37	66.67	66.86	67.02	67.56	63.39	66.17	67.53	68.37	69.02
		F	61.02	58.50	55.01	50.24	43.76	39.60	51.39	57.22	59.63	60.69	52.91	53.71	53.92	53.87	54.12	35.92	51.01	57.78	60.97	62.84
	HIGH	P	79.33	76.28	68.88	58.08	45.19	44.34	61.41	70.76	74.75	76.49	66.72	66.48	65.86	64.96	63.73	47.47	63.10	69.68	72.79	74.72
		R	80.14	80.14	79.65	78.59	76.99	76.25	78.82	79.85	80.22	80.37	78.87	79.06	79.16	79.21	79.22	76.63	78.83	79.68	80.05	80.32
		F	79.52	76.97	70.71	61.29	49.79	48.56	64.18	72.41	75.83	77.31	68.60	68.40	67.90	67.17	66.22	51.62	65.62	71.34	74.02	75.69
MF-Dispersion	LOW	P	61.34	58.24	53.50	47.27	39.28	35.36	48.82	55.96	59.07	60.44	51.23	52.04	52.20	52.03	52.14	31.97	48.48	56.37	60.25	62.57
		R	69.26	69.20	69.07	68.61	67.71	66.15	68.43	69.45	69.84	69.97	68.24	68.55	68.75	68.89	69.41	65.08	68.04	69.45	70.31	70.96
		F	63.36	60.69	56.66	51.24	44.13	40.27	52.61	58.89	61.57	62.74	54.53	55.27	55.45	55.34	55.50	37.00	52.43	59.37	62.67	64.61
	HIGH	P	77.93	75.15	68.33	58.07	45.66	44.62	61.24	70.12	73.78	75.39	66.22	65.94	65.32	64.43	63.24	47.84	62.82	68.99	71.85	73.64
		R	78.61	78.61	78.21	77.30	75.86	75.21	77.50	78.39	78.68	78.81	77.51	77.68	77.77	77.81	77.81	75.57	77.49	78.22	78.54	78.76
		F	78.08	75.74	69.94	60.95	49.85	48.48	63.73	71.56	74.71	76.08	67.88	67.65	67.13	66.42	65.48	51.62	65.07	70.44	72.93	74.50

TABLE 4 Mean values for the Precision (P), Recall (R), and F-measure (F) by each search parameter value and each MD-MFLP.

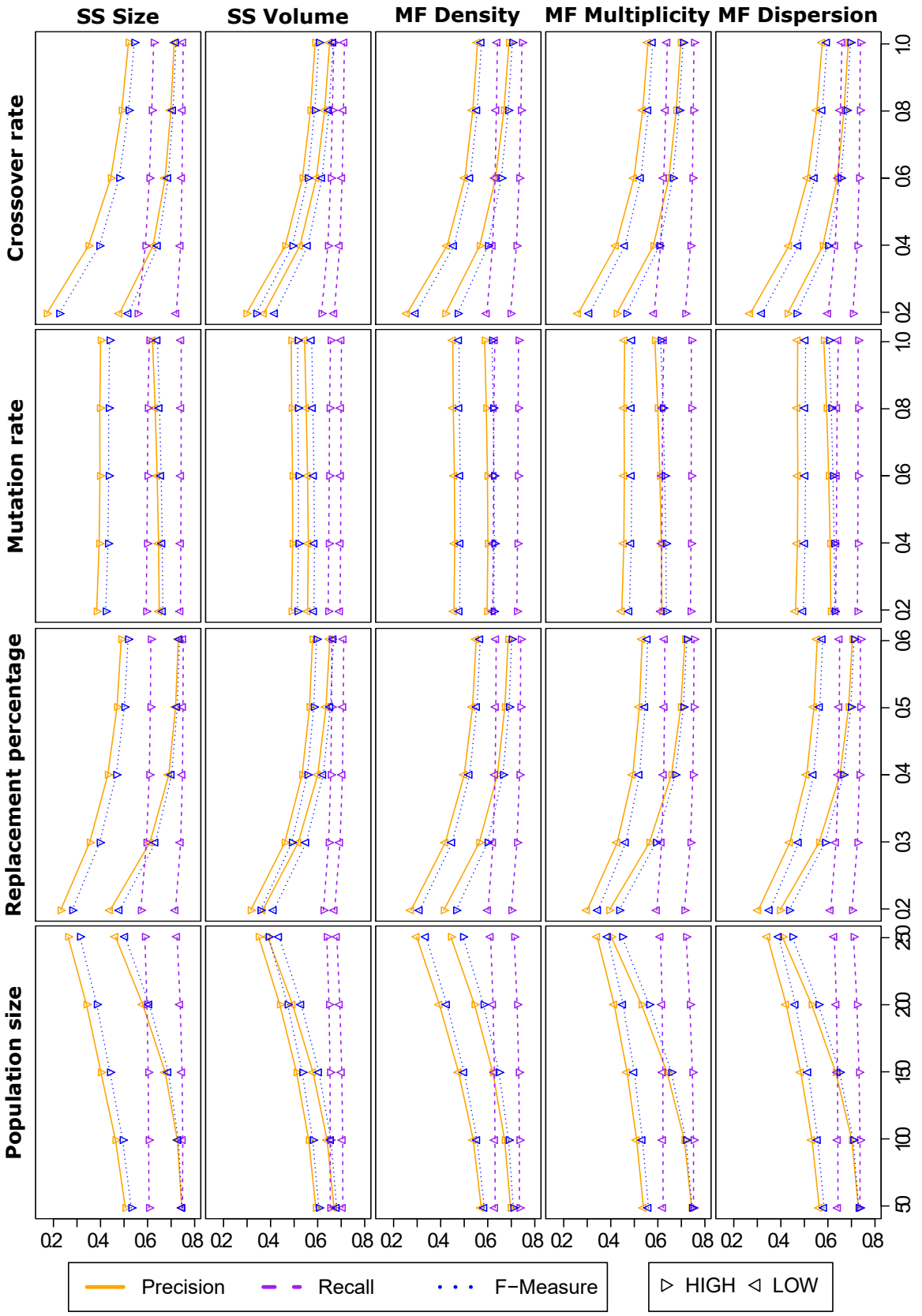


FIGURE 8 Graphs of the Precision, Recall, and F-Measure values obtained by each MD-MFLP (HIGH and LOW) and each search parameter.

Ranking	Search Space		Model Fragment			Parameters				Performance		
	Size	Volume	Density	Multiplicity	Dispersion	λ	μ	p_m	p_c	P	R	F
1	LOW	LOW	HIGH	LOW	LOW	50	50%	1.0	1.0	86.60	86.60	86.60
2	LOW	LOW	HIGH	LOW	LOW	100	30%	1.0	1.0	86.56	86.56	86.56
3	LOW	LOW	HIGH	LOW	LOW	200	40%	0.6	1.0	86.55	86.55	86.55
4	LOW	LOW	HIGH	LOW	LOW	200	60%	1.0	1.0	86.55	86.55	86.55
5	LOW	LOW	HIGH	LOW	LOW	150	40%	0.4	1.0	86.51	86.51	86.51
					...							
14996	HIGH	LOW	LOW	LOW	LOW	200	20%	0.2	0.2	2.30	48.63	4.37
14997	HIGH	LOW	LOW	LOW	LOW	250	20%	0.4	0.2	2.25	51.51	4.29
14998	HIGH	LOW	LOW	LOW	LOW	250	20%	0.2	0.2	2.23	51.21	4.26
14999	HIGH	LOW	LOW	LOW	LOW	250	20%	0.8	0.2	2.21	50.01	4.21
15000	HIGH	LOW	LOW	LOW	LOW	250	20%	0.6	0.2	2.21	50.30	4.21

TABLE 5 Search parameter vector and MD-MFLP performance ranking ordered by F-measure.

each MD-MFLP in order to determine the statistically significant differences among the results of each pair of search parameter values and of each MD-MFLP. To do that, we used the Tukey HSD (Honestly Significant Difference)[24] post hoc test, which is designed to test all of the means against each other pairwise. In other words, each mean is compared, in turn, with every other mean. Table 6 shows the $p - values$ of the Tukey HSD test comparing the different values for each search parameter and for each MD-MFLP. In addition, the $diff$ value shows the difference in the means of the two groups.

The largest differences for the population size parameter were between the values of $\mu_5 = 250$ and $\mu_1 = 50$. For the replacement parameter, they were between the values of $\lambda_5 = 60\%$ and $\lambda_1 = 20\%$. For the mutation rate parameter, they were between the values of $p_{m2} = 0.4$ and $p_{m5} = 1.0$, and, for the crossover rate parameter, they were between the values of $p_{c1} = 0.2$ and $p_{c5} = 1.0$. With regard to the MD-MFLP, the SS-Size showed the biggest differences between HIGH and LOW values.

4.2.2 | Effect size

Apart from determining if two search parameter values and two MD-MFLPs are significantly different in performance, it is also important to assess the magnitude of those differences. To do so, we used effect size measures; specifically, we applied the Cohen d value[25].

Table 7 contains the Cohen d values for each search parameter and for each MD-MFLP. Cohen d values above 0.8 mean high interaction effect (colored gray), values from 0.5 to 0.8 mean medium interaction effect, and values from 0.2 to 0.5 mean low interaction effect. Values from 0 to 0.2 mean almost no interaction effect.

The Cohen d values of population size, replacement percentage, and crossover rate parameters affirm that those search parameters have high interaction effect in precision and F-measure but not in recall. However, the mutation rate parameter did not have interaction in any of the performance measures. In contrast, all MD-MFLPs had a higher effect in recall than in precision. The SS-Size measure had a high effect in precision, recall, and F-measure.

4.3 | Findings and discussion

The evaluation presented in this paper provides a detailed analysis of the performance obtained by the results of the study of different search parameters of a search algorithm to locate 1,895 MFLPs extracted from two different industrial domains. The results obtained in the evaluation allow us to answer each of the research questions.

RQ1. Is there any impact on performance regarding the search parameter vector for model fragment location problems?

The population size, replacement percentage, and crossover rate parameters have large effects on the precision value while the mutation rate parameter has a small effect. The population size parameter and the precision value are inversely proportional. In other words, the precision value increases as the population size parameter decreases. The replacement percentage parameter

			Tukey HSD					
			Precision		Recall		F-measure	
			$diff$	p	$diff$	p	$diff$	p
Search parameters	Population size (μ)	100 - 50	2.96	0.00	0.03	0.97	2.53	0.00
		150 - 50	8.60	0.00	0.28	0.00	7.32	0.00
		200 - 50	16.56	0.00	0.93	0.00	14.27	0.00
		250 - 50	26.44	0.00	2.06	0.00	23.09	0.00
		150 - 100	5.63	0.00	0.25	0.00	4.78	0.00
		200 - 100	13.59	0.00	0.90	0.00	11.73	0.00
		250 - 100	23.47	0.00	2.03	0.00	20.56	0.00
		200 - 150	7.96	0.00	0.65	0.00	6.95	0.00
		250 - 150	17.84	0.00	1.78	0.00	15.77	0.00
		250 - 200	9.88	0.00	1.13	0.00	8.82	0.00
	Replacement percentage (λ)	0.3 - 0.2	14.82	0.00	2.29	0.00	13.58	0.00
		0.4 - 0.2	22.70	0.00	3.25	0.00	20.53	0.00
		0.5 - 0.2	26.05	0.00	3.59	0.00	23.41	0.00
		0.6 - 0.2	27.52	0.00	3.72	0.00	24.67	0.00
		0.4 - 0.3	7.88	0.00	0.96	0.00	6.95	0.00
		0.5 - 0.3	11.23	0.00	1.31	0.00	9.83	0.00
		0.6 - 0.3	12.70	0.00	1.44	0.00	11.08	0.00
		0.5 - 0.4	3.35	0.00	0.34	0.00	2.88	0.00
		0.6 - 0.4	4.82	0.00	0.48	0.00	4.14	0.00
		0.6 - 0.5	1.47	0.00	0.13	0.03	1.26	0.00
	Mutation rate (p_m)	0.4 - 0.2	0.35	0.00	0.25	0.00	0.33	0.00
		0.6 - 0.2	0.17	0.26	0.40	0.00	0.21	0.06
		0.8 - 0.2	0.31	0.00	0.50	0.00	0.16	0.23
		1.0 - 0.2	0.75	0.00	0.80	0.00	0.47	0.00
		0.6 - 0.4	0.18	0.24	0.15	0.01	0.12	0.54
		0.8 - 0.4	0.66	0.00	0.25	0.00	0.49	0.00
		1.0 - 0.4	1.10	0.00	0.54	0.00	0.80	0.00
		0.8 - 0.6	0.48	0.00	0.10	0.15	0.37	0.00
		1.0 - 0.6	0.93	0.00	0.40	0.00	0.68	0.00
		1.0 - 0.8	0.45	0.00	0.29	0.00	0.31	0.00
	Crossover rate (p_c)	0.4 - 0.2	15.85	0.00	2.51	0.00	14.58	0.00
		0.6 - 0.2	23.00	0.00	3.63	0.00	20.86	0.00
		0.8 - 0.2	26.45	0.00	4.25	0.00	23.81	0.00
		1.0 - 0.2	28.55	0.00	4.73	0.00	25.59	0.00
		0.6 - 0.4	7.15	0.00	1.12	0.00	6.28	0.00
		0.8 - 0.4	10.59	0.00	1.75	0.00	9.22	0.00
		1.0 - 0.4	12.69	0.00	2.22	0.00	11.01	0.00
		0.8 - 0.6	3.44	0.00	0.63	0.00	2.95	0.00
		1.0 - 0.6	5.54	0.00	1.10	0.00	4.73	0.00
		1.0 - 0.8	2.10	0.00	0.47	0.00	1.78	0.00
MD-MFLP	SS-Size	LOW - HIGH	24.33	0.00	14.19	0.00	22.33	0.00
	SS-Volume	LOW - HIGH	6.35	0.00	4.90	0.00	6.54	0.00
	MF-Density	LOW - HIGH	13.95	0.00	10.24	0.00	14.32	0.00
	MF-Multiplicity	LOW - HIGH	15.04	0.00	12.21	0.00	13.95	0.00
	MF-Dispersion	LOW - HIGH	13.10	0.00	8.95	0.00	11.70	0.00

TABLE 6 Tukey HSD test values for each search parameter and for each MD-MFLP

		Cohen d value		
		Precision	Recall	F-Measure
Search parameters	Population size (μ)	0.964	0.130	0.916
	Replacement percentage (λ)	0.971	0.235	0.948
	Mutation rate (p_m)	0.037	0.034	0.030
	Crossover rate (p_c)	1.005	0.301	0.980
MD-MFLP	SS-Size	0.901	0.997	0.913
	SS-Volume	0.216	0.312	0.245
	MF-Density	0.485	0.680	0.553
	MF-Multiplicity	0.525	0.831	0.537
	MF-Dispersion	0.454	0.586	0.445

TABLE 7 Cohen d values for each search parameter and for each MD-MFLP.

and the crossover rate parameter are directly proportional to the precision value, i.e., the precision increases as they increase. However, all search parameters had small effects on the recall value.

The results shows that the best mean performance were obtained by population size $\mu_3 = 150$, replacement percentage $\lambda_4 = 50\%$, mutation rate $p_{m5} = 1.0$, and crossover rate $p_{c5} = 1.0$. If we consider the search parameters separately, the population size value, the replacement percentage, and mutation rate vary to achieve the best performance. However, the crossover value remained at $p_{c5} = 1.0$.

Our statistical analysis of the results confirms that the crossover rate parameter is the one that presents larger differences and higher effect size. The differences observed in the results when switching the value for crossover rate are significant and yield large differences in the performance. Higher values of crossover rate yield to higher values of performance (around a 25% improvement in F-measure).

RQ2. Is there any impact on performance regarding the measures that define the model fragment location problems?

There were differences in the performance metrics between HIGH and LOW values of the MD-MFLP. The SS-Size measure obtained the highest differences for precision and recall. The MF-Multiplicity had a large effect on the recall value but a medium effect on precision value. MF-Density and MF-Multiplicity had medium effects on precision and recall values. The SS-Volume measure obtained the smallest differences for precision and recall.

The measures that are related with the search space (SS-Size and SS-Volume) obtained the best values in performance with the LOW values. However, the measures that are related to the solution (MF-Density, MF-Multiplicity, and MF-Dispersion) obtained the best values in performance with the HIGH values. The localization problems used in the evaluation have been classified using the five MD-MFLPs measures analyzed. This information can be very useful for other software engineers facing search problems in order to decide which parameters to use in their evolutionary algorithms.

RQ3. Are there any relations between the measures that define the model fragment location problems and the search parameter vector used that affect the performance?

There are interactions between the search parameter vector and the MD-MFLP that affect the performance metrics. The highest interaction between the MD-MFLP and the search parameter vector is produced by different values of SS-Size. The most important parameter value is the crossover rate, which maintains its value in many of the scenarios.

In addition, this evaluation allowed us to realize that the nature of the search problems has a greater effect on recall, while the search strategy has a greater effect on the precision of the solutions found.

4.4 | Threats to validity

This section presents some threats to the validity of the results presented. We have followed the guidelines suggested by De Oliveira et. al[26].

Conclusion validity: We identified three threats of this type. The first threat is not accounting for random variation. To address this threat, we considered 100 independent runs for each of the test cases for each of the search parameter vectors. The

second threat is the lack of formal hypotheses and statistical tests. In this work, we employed standard statistical analysis following accepted guidelines[22] to avoid this threat. The third threat is the lack of good descriptive analysis. In this work, we have used precision, recall, and F-measure metrics to analyze the confusion matrix obtained from the experiments; however, other metrics could be applied.

Internal validity: We identified two threats of this type. The first identified threat of this type is the poor search parameter settings. In this work, we evaluated which search parameter values worked best when performing model fragment location with our algorithm. In addition, the choice of the k value in the application of SVD can produce sub-optimal accuracy when using LSI for software artifacts[27]. The second threat is the lack of real problem instances. The evaluation of this paper was applied to 1,895 location problems from two industrial case studies, BSH and CAF.

Construct validity: We identified one threat of this type. The identified threat is the lack of assessing the validity of cost measures. To address this threat, we performed a fair comparison among the algorithms with different search parameter values by generating the same number of model fragments and allocating the same budget time.

External validity: We identified two threats of this type. The first is the lack of a clear object selection strategy and the second is the lack of evaluations for instances of growing size and complexity. To mitigate these threats, we used a large number of case studies from two industrial partners (BSH and CAF). Our instances are extracted from real-world problems. Also, the approach was evaluated in two different domains that varied in size and complexity.

5 | RELATED WORK

Some works have investigated the effect of the use of different operators that are tailored to the domain on the results of the search being performed[28,29,30,31]. For instance,[30] extends their previous work by defining software architectures[32] to analyze the effect of the crossover operator in genetic algorithms that are used to synthesize software architecture designs. The authors compare sexual and asexual crossover operators, applying them to two test cases to conclude that the asexual crossover (i.e., no crossover) provides better results for that domain than a regular sexual crossover. However, the work was further refined[31] to propose a complementary crossover that is capable of yielding solutions with better quality than the asexual crossover. The crossover operator proposed can find complementary parents and produce offspring that combine the best from both parents.

Similarly,[28] introduces a feature-driven crossover operator that is capable of providing better results when applied to optimize a product line architecture. The study compares a multi-objective evolutionary algorithm that is applied to two case studies using their crossover operator or no crossover operator at all.

Harman et al.[29] propose a new crossover operator to be applied in the context of automated software re-modularization. The new crossover operator aims to preserve building blocks from parents that are transferred without modifications to the new offspring. This leads to better results than a regular crossover operator.

These works can improve the results by using a different operator rather than by changing the values of the parameters of the operator. The different operators that are applied by the evolutionary algorithm in those works correspond to qualitative parameters. In contrast, the focus in our work are the quantitative parameters of the evolutionary algorithm, looking for the values that provide the best results with current genetic operators, which leads to great improvement.

Some works focus on the tuning of the parameters of evolutionary algorithms[33,16,34]. For instance,[33] presents a survey on parameter tuning and parameter control and discusses several techniques to achieve it. The authors also make the distinction between parameter tuning (how to choose parameters before running the search algorithm) and parameter control (how to change parameters while the search is being performed). However, it is important to note that parameter control does not fully address the problems of parameter tuning as the introduction of the control mechanisms usually leads to more parameters that need to be set.

The work from[16] deals directly with the *No Free Lunch theorem*[35] (it is impossible to tune a search algorithm so that it will have optimal parameter values for all problem instances). The authors perform a large empirical analysis in the context of test data generation for object-oriented software to determine the impact of the tuned parameters on the searches. They conclude that parameter tuning affects the performance of search algorithms; however, well-tuned parameters are complex to find, and default values may be enough.

In[34], the authors carry out a more general study of the parameter tuning of evolutionary algorithms. Contrary to[16], they conclude that by using tuning algorithms one cannot only obtain superior parameter values, but also a lot of information about problem instances, parameter values, and algorithm performance.

Nevertheless, those works are applied to the (more general) field of Search-Based Software Engineering and therefore do not take the particularities of the SBMDE field into account. It is not clear whether search parameter values that provide good results in problem instances that do not deal with models will behave similarly with SBMDE problem instances.

6 | CONCLUSION

More and more, researchers are reformulating MDE activities as search problems. These works use Search-Based optimization techniques (mainly those from the evolutionary computation literature) to automate the search for optimal and near-optimal solutions. However, these works are neglecting the influence of the search parameter values selected and the nature of their problems in their results.

In this work, we have performed an evaluation to determine the impact of different search parameter values when performing model fragment location following an evolutionary algorithm and taking into account the nature of the problems to locate. It turns out that there are interactions between the search parameter vector and the MD-MFLP that affect the performance metrics. Different values of population size, replacement percentage, or crossover rate parameters produce variations of around 30% in performance, but the mutation rate parameter produces differences of less than 5% in performance. In addition, LOW values related to the search space and HIGH values related to the model fragment in the MD-MFLPs obtain better values in performance. To achieve these results, the approach has been tested using 625 different search parameter vectors applied to 1,895 different MFLPs. In addition, the results have been supported with a statistical analysis that determines that the results are significant and are not due to mere chance.

Finally, we think that the SBMDE community could benefit from the result of our work by taking into account the influence of search parameter values and the nature of the MFLPs in their results.

References

1. Boussaïd I, Siarry P, Ahmed-Nacer M. A survey on search-based model-driven engineering. *Automated Software Engineering* 2017; 24(2): 233–294. doi: 10.1007/s10515-017-0215-4

2. Kent S. Model Driven Engineering. In: Butler M, Petre L, Sere K. , eds. *Integrated Formal Methods*Springer Berlin Heidelberg; 2002; Berlin, Heidelberg: 286–298.

3. Harman M, Jones BF. Search-based software engineering. *Information and software Technology* 2001; 43(14): 833–839.

4. Arcega L, Font J, Haugen Ø, Cetina C. An approach for bug localization in models using two levels: model and metamodel. *Softw. Syst. Model.* 2019; 18(6): 3551–3576. doi: 10.1007/s10270-019-00727-y

5. Pérez F, Lapeña R, Font J, Cetina C. Fragment retrieval on models for model maintenance: Applying a multi-objective perspective to an industrial case study. *Inf. Softw. Technol.* 2018; 103: 188–201.

6. Echeverría J, Font J, Pérez F, Cetina C. Comparison of Search Strategies for Feature Location in Software Models. *J. Syst. Softw.* 2021; 181(C). doi: 10.1016/j.jss.2021.111037

7. Font J, Arcega L, Haugen Ø, Cetina C. Building software product lines from conceptualized model patterns. In: ; 2015: 46–55

8. Font J, Arcega L, Haugen Ø, Cetina C. Achieving Feature Location in Families of Models through the use of Search-Based Software Engineering. *IEEE Transactions on Evolutionary Computation* 2017; PP(99): 1-1. doi: 10.1109/TEVC.2017.2751100

9. Font J, Arcega L, Haugen Ø, Cetina C. Handling nonconforming individuals in search-based model-driven engineering: nine generic strategies for feature location in the modeling space of the meta-object facility. *Software and Systems Modeling* 2021. doi: 10.1007/s10270-021-00870-5

10. Ballarín M, Marcén AC, Pelechano V, Cetina C. Measures to Report the Location Problem of Model Fragment Location. In: MODELS '18. ACM; 2018; New York, NY, USA: 189–199.

11. Manning CD, Raghavan P, Schütze H, others . *Introduction to information retrieval*. Cambridge university press Cambridge . 2008.

12. Manning C, Schutze H. *Foundations of Statistical Natural Language Processing*. MIT Press . 1999.

13. Landauer TK, Foltz PW, Laham D. An introduction to latent semantic analysis. *Discourse processes* 1998; 25(2-3): 259–284.

14. Font J, Arcega L, Haugen Ø, Cetina C. Feature Location in Models Through a Genetic Algorithm Driven by Information Retrieval Techniques. In: MODELS '16. ACM; 2016; New York, NY, USA: 272–282

15. Eiben AE, Smit SK. *Evolutionary Algorithm Parameters and Methods to Tune Them*: 15–36; Berlin, Heidelberg: Springer Berlin Heidelberg . 2012

16. Arcuri A, Fraser G. Parameter tuning or default values? An empirical investigation in search-based software engineering. *Empirical Software Engineering* 2013; 18(3): 594–623. doi: 10.1007/s10664-013-9249-9

17. Sayyad AS, Goseva-Popstojanova K, Menzies T, Ammar H. On Parameter Tuning in Search Based Software Engineering: A Replicated Empirical Study. In: RESER '13. IEEE Computer Society; 2013; USA: 84–90

18. Font J, Arcega L, Haugen Ø, Cetina C. Feature Location in Model-Based Software Product Lines Through a Genetic Algorithm. In: ; 2016: 39–54

19. Nam J, Kim S. CLAMI: Defect Prediction on Unlabeled Datasets (T). In: ; 2015: 452-463

20. MacCallum RC, Zhang S, Preacher KJ, Rucker DD. On the practice of dichotomization of quantitative variables.. *Psychological methods* 2002; 7(1): 19.

21. Stehman SV. Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment* 1997; 62(1): 77–89. doi: 10.1016/S0034-4257(97)00083-7

22. Arcuri A, Briand L. A Hitchhiker's Guide to Statistical Tests for Assessing Randomized Algorithms in Software Engineering. *Softw. Test. Verif. Reliab.* 2014; 24(3): 219–250. doi: 10.1002/stvr.1486

23. Tabachnick BG, Fidell LS. *Experimental designs using ANOVA*. 724. Thomson/Brooks/Cole Belmont, CA . 2007.

24. Tukey JW. Comparing individual means in the analysis of variance.. *Biometrics* 1949; 5 2: 99-114.

25. Cohen J. Statistical power analysis. *Current directions in psychological science* 1992; 1(3): 98–101.

26. Oliveira Barros dM, Dias-Neto AC. 0006/2011-Threats to Validity in Search-based Software Engineering Empirical Studies. *RelaTe-DIA* 2011; 5(1).

27. Panichella A, Dit B, Oliveto R, Penta MD, Poshyvanyk D, Lucia AD. Parameterizing and Assembling IR-Based Solutions for SE Tasks Using Genetic Algorithms. In: . 1. ; 2016: 314-325

28. Colanzi TE, Vergilio SR. A feature-driven crossover operator for multi-objective and evolutionary optimization of product line architectures. *Journal of Systems and Software* 2016; 121: 126 - 143. doi: https://doi.org/10.1016/j.jss.2016.02.026

29. Harman M, Hierons RM, Proctor M. A New Representation And Crossover Operator For Search-based Optimization Of Software Modularization. In: ; 2002: 1351–1358.

30. Räihä O, Koskimies K, Mäkinen E. Empirical study on the effect of crossover in genetic software architecture synthesis. In: ; 2009: 619-625

31. Räihä O, Koskimies K, Mäkinen E. Complementary crossover for genetic software architecture synthesis. In: ; 2010: 266-271

32. Räihä O, Koskimies K, Mäkinen E, Systa T. Pattern-based Genetic Model Refinements in MDA. *Nordic J. of Computing* 2008; 14(4): 338–355.

33. Eiben AE, Michalewicz Z, Schoenauer M, Smith JE. Parameter control in evolutionary algorithms. In: Springer. 2007 (pp. 19–46).

34. Eiben A, Smit S. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation* 2011; 1(1): 19 - 31. doi: https://doi.org/10.1016/j.swevo.2011.02.001

35. Wolpert DH, Macready WG. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1997; 1(1): 67-82. doi: 10.1109/4235.585893

AUTHOR BIOGRAPHY

Isis Roca. is researcher in the SVIT Research Group (https://svit.usj.es) at Universidad San Jorge, Zaragoza, Spain. She is PhD student in Computer Science at the Polytechnic University of Valencia, Spain. She received the M.Sc. degree in advanced software technologies from Universidad San Jorge. Her current research interest include Model Driven Engineering, Search-Based Software Engineering and model fragment localization.

Jaime Font. Jaime Font received the Ph.D. degree in computer science from University of Oslo, Oslo, Norway. He is Associate Professor with the SVIT Research Group, Universidad San Jorge, Zaragoza, Spain. His current research interests include reverse engineering, evolutionary computation, variability modeling, and procedural content generation. He publishes his research results and participates in high-level international software engineering conferences and journals, such as IEEE Transactions on Evolutionary Computation (TEVC), and Software and System Modeling (SoSyM) journal.

Lorena Arcega. She is Tenure Track Professor with the SVIT Research Group, Universidad San Jorge, Zaragoza, Spain. She received a Ph.D. degree in computer science from the University of Oslo, Oslo, Norway. Her current research interests include models at runtime, software maintenance and evolution, and variability modeling. She publishes her research results and participates in high-level international software engineering conferences and journals, such as the International Conference on Model-Driven Engineering Languages and Systems (MODELS), and Software and System Modeling (SoSyM) journal.

Carlos Cetina. He is Associate Professor with San Jorge University and the Head of the SVIT Research Group. He received a Ph.D in computer science from the Polytechnic University of Valencia. His research focuses on software product lines and model-driven development. His research results have reshaped software development in world-leading industries from heterogeneous domains ranging from induction hob firmware to train control and management systems. More information about his background can be found at his website: http://carloscetina.com