Journal of Object Technology
Published by AITO — Association Internationale pour les Technologies Objets
http://www.jot.fm/

Searching for Optimal Models: Comparing Two Encoding Approaches

Stefan John[a] Alexandru Burdusel[b] Robert Bill[c]
Daniel Strüber[d] Gabriele Taentzer[a] Steffen Zschaler[b]
Manuel Wimmer[e]

a. Philipps-Universität Marburg, Marburg, Germany
b. King's College London, London, United Kingdom
c. TU Wien | Austrian Center for Digital Production, Wien, Austria
d. Chalmers University | University of Gothenburg, Gothenburg, Sweden
e. CDL-MINT, Johannes Kepler University, Linz, Austria

Abstract Search-Based Software Engineering (SBSE) is about solving software development problems by formulating them as optimization problems. In the last years, combining SBSE and Model-Driven Engineering (MDE), where models and model transformations are treated as key artifacts in the development of complex systems, has become increasingly popular. While search-based techniques have often successfully been applied to tackle MDE problems, a recent line of research investigates how a model-driven design can make optimization more easily accessible to a wider audience. In previous model-driven optimization efforts, a major design decision concerns the way in which solutions are encoded. Two main options have been explored: a model-based encoding representing candidate solutions as models, and a rule-based encoding representing them as sequences of transformation rule applications. While both encodings have been applied to different use cases, no study has yet compared them systematically. To close this gap, we evaluate both approaches on a common set of optimization problems, investigating their impact on the optimization performance. Additionally, we discuss their differences, strengths, and weaknesses laying the foundation for a knowledgeable choice of the right encoding for the right problem.

Keywords Model-driven Engineering Search-based Software Engineering Optimization Encoding Comparative evaluation

1 Introduction

Many software engineering problems give rise to a tremendous space of possible solutions that differ in various qualities, such as their performance, resource efficiency, and understandability. To efficiently find optimal solutions, *search-based software engineering (SBSE)* [HJ01] seeks to formulate the problem as an optimization problem over one or multiple fitness functions capturing the qualities of interest. By using metaheuristic search techniques, the available solution space can be explored efficiently. Due to their generality, a technique of particular relevance are genetic algorithms [HMZ12], which use the evolutionary operators of mutation, crossover, and selection to perform a guided search over the search space.

Model-driven engineering (MDE) is a paradigm that aims to raise the level of abstraction in a broad range of application domains by the use of models, which are continuously refined and transformed. Recently, research combining SBSE and MDE for a range of purposes has become increasingly popular. The term *search-based model-driven engineering* (SBMDE, [BSAN17]) has been proposed as an umbrella term for these efforts. One particular line of research in SBMDE, which we call *model-driven optimization* (MDO), aims to reduce the level of expertise required by users of SBSE techniques[1]. In MDO, models are used to specify optimization problems and transformation rules are used to explore the search space. Thus, rather than becoming involved in the intricacies of the used optimization technology, users interact with a domain-specific formulation of their problem. They can rely on the familiar modeling and model transformation tools to inspect the solutions and specify the change operations.

Recently, a variety of MDO frameworks has emerged [BFT$^+$17, AVS$^+$14, ZM16, Str17] and been applied successfully in numerous use-cases, including security-oriented software refactoring [RKL$^+$18], model generation [SNV18], transformation modularization [FTK$^+$17], and various more examples [BSAN17]. A key distinction in MDO frameworks concerns the way in which solutions are encoded [ZM16]: The *model-based encoding* approach represents solutions as models. In the *rule-based encoding* approach, a solution is a sequence of rule calls in the context of a given input model. Both encoding approaches have distinct advantages: The model-based approach reduces the overhead of applying transformations before the solution is evaluated. It also removes the effort for tracking detail information of the rule calls. The rule-based approach, instead of only incrementally changing the solutions, allows to go back in time easily and deviate from changes made earlier, which may allow the search to move faster through the search space.

So far, there has been no systematic assessment of how the choice of encoding impacts the performance of the search. In existing use-cases, the approach was chosen in an ad-hoc manner, based on the availability of a specific MDO framework. Systematic evidence for the suitability of the chosen approach may help developers in selecting a solution that best fits their use-case and lower the acceptance threshold for MDO in practice.

In this paper, we aim to compare the two main encoding approaches in MDO frameworks. We study the implementation of these approaches in two state-of-the-art MDO frameworks that differ in the encoding approach used, but otherwise share the same technological basis. This setup allows us to attribute any observed differences in performance to the used encoding. Specifically, we consider MOMoT

[1] As opposed to applying SBSE techniques to solve MDE problems.

[BFT$^+$17] and MDEOptimiser [BZS18], which follow the rule-based and the model-based approach, respectively. Both are built atop of the EMF modeling platform [SBMP08], the Henshin model transformation language [ABJ$^+$10, SBG$^+$17], and the MOEA evolutionary search framework [MOE]. Problems are specified by metamodels; model transformations are performed by Henshin transformation rules.

The main contributions of the paper are as follows:

1. A *qualitative comparison* between the model-based and the rule-based encoding in MDO frameworks, based on a systematic study of their features.
2. A *quantitative comparison* of both encodings with their implementations in MOMoT and MDEOptimiser, based on their performance (regarding solution quality and execution time) in a set of three diverse use cases.
3. *Insights into the applicability* of both encoding approaches; their strengths and weaknesses. We study whether the differences can be attributed to the different encoding approaches.

The paper is structured as follows: Section 2 introduces the use-cases considered in this paper. Section 3 describes MDO. Section 4 and 5 are devoted to the qualitative and quantitative comparative evaluation, respectively. Section 6 provides interpretations for the observed results, while Section 7 points out threats to their validity. Section 8 discusses related work. Section 9 concludes the paper.

2 Optimization Problems

For our experimental analysis, we focus on three combinatorial optimization problems. Such problems can be described by providing [GJ79]: (i) a problem domain; (ii) problem instances, each with a finite set of candidate solutions; (iii) a function which maps each candidate solution to a rational number.

Combinatorial optimization problems are often encountered in fields such as engineering, software engineering and finance [CCRS04]. In contrast to other optimization categories (e.g. Integer Programming, Linear Programming), where sets of equations need to be solved, requirements of combinatorial problems are usually formulated by objects and their relationships. This makes them ideal candidates for Model-driven Optimization.

2.1 Problem Descriptions

In this section, we briefly introduce the case studies we will use for quantitative evaluation. We provide only a rough description of the key features of each case study; more detail can be obtained from the websites linked in the footnotes.

Class Responsibility Assignment (CRA) The CRA use case [BBL10] stems from the domain of software design and, initiated by the *Transformation Tool Contest 2016 (TTC'16)* [FTCW16], has been addressed by several works in the last years [FSK17, BZ18, Str17]. A software system is defined by a set of features (methods and attributes) and dependencies. Dependencies can be functional, i.e., one method calling another one, or data-driven, i.e., an attribute is processed by a method. Classes can be added to encapsulate features and modularize the system. With the constraint of assigning all features to classes, the optimization goal is to reach a good modular software design. To assess the quality of such a class diagram, the *CRA-Index* is used, which aggregates the metrics of cohesion and coupling. To increase the maintainability and comprehension of a system, combinations of high cohesion and low coupling, reflected by higher CRA-Index values, are desirable [BDW98]. To facilitate easier comprehension of the model changes needed to reach a modular design, we define an extended CRA case (CRA ext.) with the additional objective to minimize the number of transformations performed by the optimization process.

Next Release Problem (NRP) The NRP is about planning which features to include in the next release of a software product. Features are requested by customers in terms of requirements and are implemented as software artifacts. The two optimization objectives are to minimize the development costs, by developing as few artifacts as possible, and to maximize customer satisfaction, by fulfilling requirements. Typical versions of the problem [DZA+11] do not allow nested requirements, partial fulfillment of requirements, or dependencies between software artifacts. Based on the work of Burton and Poulding [BP13], our problem specification[2] considers these additional facets.

Refactoring (REF) The REF use case[3] has been taken from the TTC'13. Program refactoring is a common task in agile development typically performed manually. Developers strive to reduce the complexity and increase the comprehensibility of their programs without affecting their function. While there is a great variety of refactor operations possible, we restrict ourselves to attribute location changes to not require any control flow information about the program. The objective is to minimize the number of elements (classes and attributes), by moving duplicate attributes to existing or newly generated superclasses. In an extended version (REF ext.) we want the refactoring itself to be as simple as possible to ease manual review. Therefore, we additionally minimize the total number of refactoring operations.

2.2 Coverage of Selected Use Cases

With the selection of the above use cases, we seek to cover a wide range of optimization characteristics (Table 1) which might influence both encoding approaches differently. We selected single- as well as multi-objective problems. In particular, problems where the number of applied transformations needs to be minimized are, hypothetically, easier to address by a rule-based encoding. NRP includes an objective (maximize customer satisfaction) which guides the optimization only in a coarse manner, i.e., multiple transformations might be needed to change a solution's fitness with regard to that objective. The CRA case comprises an additional constraint. With regard to transformation rules, the use cases differ widely. The CRA case uses atomic rules which perform only small changes and contain basic rule elements. In the other cases, more complex rule structures (e.g., control flow elements) can be found. Their rules also allow larger parts of a problem instance to be restructured in one mutation. In terms of their structure, optimization problems differ in whether or not instances of types may be removed and created. While CRA and REF contain such *mutable types* (e.g. classes), NRP only allows to select or deselect existing type instances.

[2]see https://mde-optimiser.github.io/case-studies/nrp/
[3]see http://martin-fleck.github.io/momot/casestudy/class_restructuring/

	CRA	CRA ext.	NRP	Ref	Ref ext.
objective	single	multi	multi	single	multi
min. transf.	no	yes	no	no	yes
coarse obj.	no	no	yes	no	no
constrained	yes	yes	no	no	no
mut. changes	small	small	large	large	large
mut. complexity	low	low	medium	high	high
mutable types	yes	yes	no	yes	yes

Table 1 – Optimization characteristics covered by the considered use cases.

Covering these aspects of optimization problems mitigates the risk of introducing a bias into our comparison. A systematic analysis of the impact of each aspect, however, is not in the scope of this paper and is left for future work.

Regarding problem size, we considered five models, ranging from 9 features and 14 dependencies (model A) to 160 features and 600 dependencies (model E) for CRA. For NRP, two models were used. Model A with 5 customers, 25 requirements and 63 software artifacts and model B with 25 customers, 50 requirements and 203 artifacts. Models A (19 classes, 18 attributes, 15 generalizations) and model B (6 classes, 68 attributes, 4 generalizations) were used for REF.

3 Model-driven Optimization

We now introduce the MDO approach on the example of the CRA problem and illustrate its concepts in the context of Genetic Algorithms (GAs). We stick to GAs because they are prominently used in SBMDE literature [BSAN17] and are supported by both of the compared frameworks.

3.1 Preliminaries

For a basic understanding of the optimization process, we revisit GAs. GAs [Gol89] work on a population of solutions. Each individual of the population has an assigned fitness, which represents the quality of a solution with regard to the desired optimization goals. A search is performed iteratively by evolving existing solutions. Inspired by nature, evolution commonly comprises *mutation* and *crossover*. Mutation induces changes in a single solution while crossover aims at recombining multiple solutions in the hope of generating fitter offspring. After an evolution, the population of the next iteration is selected, favoring fit solutions. This cycle continues until a predefined termination criterion (e.g. fitness threshold, number of evolutions) is met.

Both encoding implementations considered in this paper use Henshin [ABJ$^+$10, SBG$^+$17] to specify and perform model transformations. Henshin is a rule-based transformation language based on the paradigm of graph transformation. A transformation rule is visually represented as a graph in which nodes and edges are annotated with actions such as *delete*, *preserve*, *create*, and *forbid* (Fig. 5). Henshin provides an interpreter engine to apply rules to input models. Roughly, a rule is applied to an input model by finding a match of its *preserve* and *delete* elements, and performing the changes specified by the *delete* and *create* elements. The existence of *forbid* elements prevents a rule from being applied.

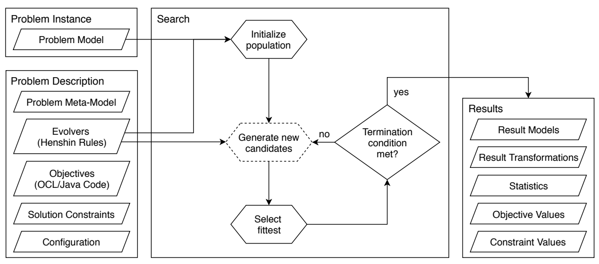
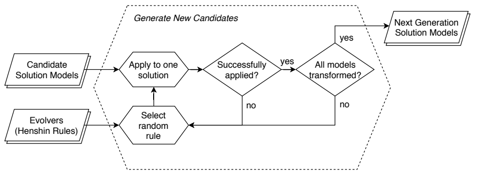


Figure 1 – General architecture overview for MDEOptimiser and MOMoT



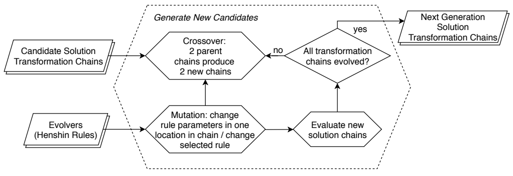

Generate New Candidates
Candidate Solution Transformation Chains
Evolvers (Henshin Rules)
Crossover: 2 parent chains produce 2 new chains
no
All transformation chains evolved?
yes
Next Generation Solution Transformation Chains
Mutation: change rule parameters in one location in chain / change selected rule
Evaluate new solution chains

Figure 2 – Overview of solution candidates generation in MDEOptimiser

Figure 3 – Overview of solution candidates generation in MOMoT

Three easily confused terms related to transformation rules need to be distinguished. First, a *rule* specifies how a transformation has to be performed and may contain formal parameters. Second, a *rule call* consists of a rule and the actual arguments used to execute the rule. And third, a rule call executed in the context of a specific model is called *transformation*.

3.2 Model-driven optimization

MDO relies on model-driven engineering concepts to specify search problems. In general, a search problem specification comprises (i) a search space description, defining all possible solutions to the problem; (ii) a method for encoding individual solutions; (iii) a set of search operators used to explore the search space; and (iv) a method for evaluating the fitness of individual solutions based on the optimization goals. In Fig. 1 we include a general overview of the architecture used by tools implementing MDO. In the following sections, we describe the core specification components.

3.2.1 Search space

In MDO, a metamodel specifies the abstract structure of a family of optimization problems; a valid model instance describes a concrete problem of that family. In addition, to describe the available search space, knowledge about immutable model parts, representing constraints of the problem at hand, is needed. Fig. 4 shows the meta-model for the CRA case. White solid elements are immutable, while colored dashed elements may be created or removed.

3.2.2 Encoding Approaches

The model-based encoding aims to reduce the computational time spent for translating solutions between the geno- and the phenotype by using the problem model itself for both. Thus, fitness can be calculated directly on newly found solutions. Fig. 2 shows an overview of the model-based approach. The rule-based approach uses a sequence of

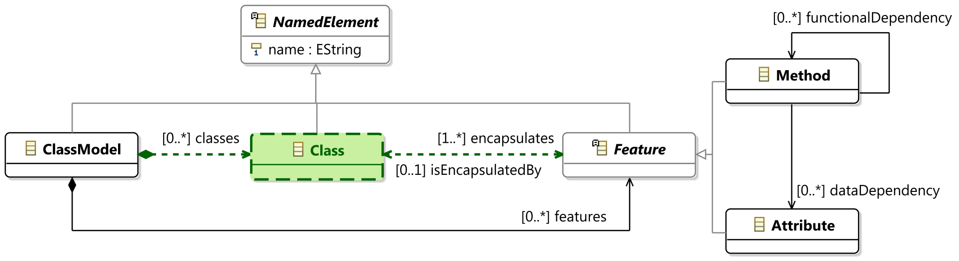


Figure 4 – Metamodel of the CRA case.

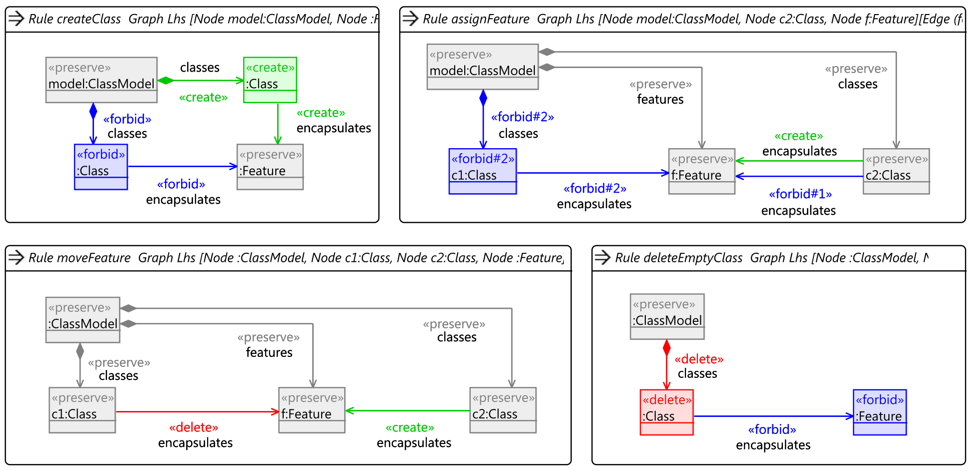


Figure 5 – Henshin transformation rules of the CRA use case.

rule calls as genotype. To evaluate the quality of a solution, its whole sequence needs to be applied to the input model before its fitness can be determined. Fig. 3 shows an overview of the rule-based approach.

3.2.3 Search Operators

In the model-based encoding, mutation is implemented in terms of applying one or multiple transformation rules to a solution model. To our knowledge, how to perform crossover to effectively recombine parts of several models remains an open research problem. For that reason, the model-based approach usually relies on mutation to explore the search space. Mutation in the rule-based approach alters the sequence of a solution. Rule calls are added, removed or changed. The approach also facilitates the use of traditional crossover operators specified for sequential encodings.

To perform model changes, in the CRA case we rely on four Henshin rules (Fig. 5), proposed by Burdusel and Zschaler [BZ16]. Unassigned features can be assigned to newly created classes by *createClass* and to existing classes by *assignFeature*. *moveFeature* relocates a feature; *deleteEmptyClass* decreases the number of available classes.

3.2.4 Fitness Functions

The quality of solution models is evaluated using fitness functions implemented in Java or OCL. Both encoding approaches allow to formulate objectives on the metamodel. In the rule-based approach, additionally, objectives on the sequence of rule calls are possible. Along with objectives, additional constraints (not covered by the metamodel) might be needed to distiguish between valid problem instances and valid solutions. It is up to the optimization algorithm used how violating solutions are treated. In the CRA case, Java classes implement the calculation of the CRA-Index, the retrieval of the number of applied rules, and the constraint of assigning all features to classes.

3.3 Implementing Tools

MOMoT as well as MDEOptimiser rely on EMF, Henshin, and the MOEA framework as their technological basis. Both offer a mature DSL to facilitate an easy configuration of optimization runs.

Both allow the user to specify the transformation rules used as mutation operators. The responsibility for generating consistent models is left to the user, who must ensure that the provided rules do not generate invalid model instances when they are applied. As MOMoT relies on the rule-based approach, rules are additionally required to match uniquely when their parameters are set. Otherwise, applying a solution sequence may lead to non-deterministic results.

As an implementation detail, MOMoT works with a fixed solution length, which needs to be specified as an additional parameter. By default, each solution of the starting population is initialized by randomly assigning rule calls to each slot of its sequence. In MDEOptimiser, following the model-based approach, solutions of the initial population are generated by applying a single random mutation to the input model provided by the user. To achieve a fair comparison, we adapted MOMoT to randomly assign a rule call to just one slot of initial solutions.

4 Qualitative Comparison

In MDO the search space is defined by (meta)models independently of the chosen encoding. The representation of solutions, however, influences most other parts of the optimization process (Table 2). In the following, we consider the main ingredients of MDO and discuss them for our two encodings.

Regarding the *choice of search operators*, the rule-based approach is more flexible than the model-based approach. Representing solutions as sequences allows the use of traditional crossover operators. Crossover is generally expected to accelerate the optimization process by recombining (potentially good) parts of already fit solutions. Additionally, the rule-based approach brings more flexibility in the choice of mutation. First, decisions of the past can be reconsidered by changing existing rule calls or entirely deleting them from a solution. Second, problems like CRA and NRP, in contrast to REF, are monotonic in the sense that their whole search space can be explored by only adding elements to (or removing them from) the initial model. For the rule-based approach, purely constructive (or purely destructive) rules are sufficient in these cases. In case of the CRA, *createClass* and *assignFeature* are such constructive rules (Fig. 5). In the model-based encoding, in contrast, additional rules are needed (e.g., *deleteEmptyClass* and *moveFeature* in CRA). As a result, evolution

	Model-based	Rule-based
problem description	initial model	initial model
solution representation	model	rule call sequence
mutation operator	rule call (sequence)	alteration of sequence slots
crossover operator	adhoc implementations	traditional variants for sequential encodings
transformation rules	both constructive and destructive rules needed	depending on use case: constructive and/or destructive rules needed
solution quality	fitness of solution model	fitness of transformation sequence and fitness of resulting model
runtime	time of applying mutation	time of applying search operators and repair plus applying sequence to initial model
consistency	metamodel conformance	metamodel conformance
completeness	depends on search rules	depends on search rules (and solution length)

Table 2 – Summary of similarities and differences of both encoding approaches.

steps are potentially spent on reverting prior transformations, slowing down the search. Additionally, more effort has to be put in rule creation.

The rule-based flexibility, however, comes at the cost of making additional repair steps necessary. Rule calls might be dependent on each other, e.g., a rule call in the CRA case creates a class to which a later rule call assigns a feature. As such, changing a single slot in a sequence of rule calls may invalidate subsequent rule calls, which needs to be addressed and may lead to a loss of evolutionary information. Compared to the model-based approach, repair steps as well as the necessity to apply a sequence to the input model to determine the fitness of the resulting model, cause a *runtime overhead* in each evolution step. Its magnitude depends on the size of the solution sequences, which is capped by the chosen solution length in MOMoT. This additional parameter also makes finding a good optimization configuration a bit harder. Note, however, that using sequences of a specific length is not a strict requirement of the rule-based approach but merely an implementation detail of MOMoT.

The rule-based encoding allows insight into how a solution has been found. It also allows more variety in formulating *objectives* as it facilitates to not only reason about the quality of a model but also of the rule call sequence generating it. We exploited that functionality in the extended CRA and REF cases, where the number of rule calls has to be minimized. In MDEOptimiser, as its encoding does not naturally support such objectives, we had to implement an additional rule call counter, effectively extending the information stored by the encoding.

Sequences of the rule-based approach do not, per se, fulfill specific *consistency constraints*. However, applying a sequence produces a solution model *consistent* with the problem metamodel. This sort of consistency is naturally given for the model-based approach. *Completeness*, i.e., whether or not the entire search space can be explored, in both approaches depends on the transformation rules provided by the user. In the rule-based approach, additional care has to be taken to choose a sufficiently large solution length when fixed size solutions are used, as solution length caps the number of changes which can be applied to an input model.

5 Quantitative Comparison

For the quantitative comparison, we are first interested in how both encoding approaches compare with regard to their optimization performance (runtime and solution quality) when the implementing tools are configured similarily. The qualitative comparison, however, shows that the rule-based approach offers configuration options for the search operators which are not present in the model-based approach. To analyze their influence on optimization performance, we also compare multiple MOMoT configurations.

We aim at answering the following research questions:

Q1: How do MDEOptimiser and MOMoT compare, in terms of optimization performance, when their behavior is aligned as much as possible?

Q2: Can the optimization performance of the rule-based approach profit from the ability to apply a traditional crossover operator?

Q3: Can the optimization performance of the rule-based approach profit from excluding destructive transformation rules?

In the following, we present the experimental setup, the results we obtained and discuss threats to the validity of our study.

5.1 Experimental Setup

For each problem instance of the use cases presented above, we performed 30 optimization runs using NSGA-II [DPAM02] a prominent representative of genetic algorithms [BSAN17] supported by both frameworks. With a population of 100 solutions each optimization instance was executed on Amazon Web Services relying on c4.2xlarge Spot instances running openjdk 1.8.0_191 and Amazon Linux release 2. Apart from these common settings, both encoding approaches differed in their parameters, runtime, and even the supported search operators.

5.1.1 Termination Condition

As a termination criterion for our experiments, we chose specific timelimits for each problem instance (Table 3). We systematically determined sensible limits by performing 10 runs with MDEOptimiser for each problem instance of each use case. The runs were stopped when no improvement took place for 100 evolutions to guarantee that a good level of convergence can be reached within the limit. The average runtime of these runs was chosen as the termination criterion for both approaches in the final experiments.

5.1.2 Operators

For both approaches, we let mutation perform exactly one change per evolution step for each solution (100% mutation rate). This is done by either executing a transformation rule (model-based) or changing the content of one sequence slot (rule-based). To answer the above research questions, we conducted experiments with different MOMoT configurations. For the direct comparison of both approaches (Q1), we configured MOMoT to use mutation only (MO). To get insights into how crossover influences the performance of the rule-based approach (Q2), another variant combining mutation with one-point crossover (MC) was used. Crossover was also applied once for each solution and step (100% crossover rate). For both approaches, we used the same set of transformation rules for each use case. To answer the last research question (Q3), in the CRA and NRP cases, we analyzed additional MOMoT configurations refraining from the use of destructive rules (e.g., *deleteEmptyClass* and *moveFeature* in the CRA case; see Fig. 5), flagging them as non-destructive (ND). In total, we tested four configurations of MOMoT as can be seen in Table 4.

To repair solutions in the rule-based approach, invalid rule calls were removed from a solution sequence.

Table 3 – Timeouts in seconds used as termination condition for each problem instance. A through E refer to the different instances of a problem. For NRP and REF, only two instances were used.

	CRA	CRA (ext.)	NRP	REF	REF (ext.)
A	5s	5s	120s	60s	800s
B	15s	15s	500s	800s	800s
C	30s	30s	-	-	-
D	300s	120s	-	-	-
E	2500s	1200s	-	-	-

5.1.3 Solution Length

As briefly discussed in Section 3.3, for MOMoT a fixed solution length determining the maximum number of possible rule calls of a solution needs to be chosen. For the CRA case, to allow exploration of the whole search space, one has to guarantee that each *feature* can be assigned to a separate *class*. To achieve that, the solution length must match at least twice the number of features in the specific problem instance. As the transformation rules included in NRP allow multiple *software artifacts* to be selected in one step, the situation changes in favor of a smaller solution length. Based on these considerations, we conducted preliminary experiments with solution lengths of 1x, 2x, 4x and 8x the number of key elements of the specific use case. Interestingly, both use cases behaved similar, with best values for 4x and 8x variants. We chose 8x for the final experiments as it was slightly dominant. Regarding the REF case, we did not have any expectations on the length of solutions, as the number of possible refactorings can hardly be guessed from the number of elements. Therefore, we experimented with a solution length of 10, 20, 40, 80 and 160. All lengths lead to a very similar solution quality, 160 being slighty benefitial.

5.2 Quality Criteria

We compare the encoding approaches on two criteria: their *solution quality* and their associated *runtime*. In single-objective scenarios the fitness of solutions can directly be used as a quality metric. In multi-objective optimization, this approach is generally not applicable. When objectives are conflicting, tradeoffs need to be considered, captured by the concept of *Pareto optimality*. A solution is said to be *Pareto optimal* if one of its objectives cannot be improved without degrading another objective. As a result, sets of mutually incomparable solutions, called Pareto front approximations, need to be considered. Additionally, a solution *dominates* another solution, if it is better in at least one objective and not worse in any of the others. *Capacity*, *convergence* and *diversity* are the main dimensions along which the quality of such sets is commonly assessed [JOZF14]. In our work we rely on two metrics measuring these dimensions for Pareto front approximations.

5.2.1 Ratio Of Best Solutions Found

To quantify the capacity of a solution set S we measure the *Best Solution Ratio* based on the cardinal *C1* metric presented by [HJ98]

$$BSR = \frac{|S \cap PF_{pseudo}|}{|PF_{pseudo}|} \tag{1}$$

PF_{pseudo} denotes the set of non-dominated solutions in the union of all solutions found for a particular problem instance. In other words, the best solutions found for that problem by any configuration used in the experiments.

5.2.2 Hypervolume

The Hypervolume indicator (HV) [ZBT07] is one of the most popular metrics used in multi-objective optimization [RLB15]. It measures the size of the area enclosed by a solution set with respect to a given reference point. In our experiments, for the latter we have chosen a point slighty worse than the nadir point given by all solutions found for a particular problem. This nadir point combines the worst objective values found so far in one vector. We use PF_{pseudo} to normalize HV. As such HV increases towards 1 as solution sets converge towards PF_{pseudo}. The distribution of solutions among the search space also influences the HV. Thereby, HV incorporates convergence and diversity in one measure.

5.3 Results

In the following we discuss the results of our experiments in terms of the research questions we seek to answer. Table 4 summarizes the outcome of the experiments with regard to the median quality and standard deviation reached by each configuration for each problem instance. The complete data set discussed in this section can be downloaded from [JBB$^+$].

Comparing Approaches (Q1) Both tools have been configured to have equal runtime across similar configurations. Despite this limit, due to the differences in the solution encodings and search operators, the number of algorithm steps performed by each implementation differs ([JBB$^+$]). Across most cases, MDEOptimiser is able to run more than twice the number of steps than MOMoT. In a few cases, however, most notably model E of the multi-objective CRA case, MOMoT outperforms MDEOptimiser with regard to evolution speed.

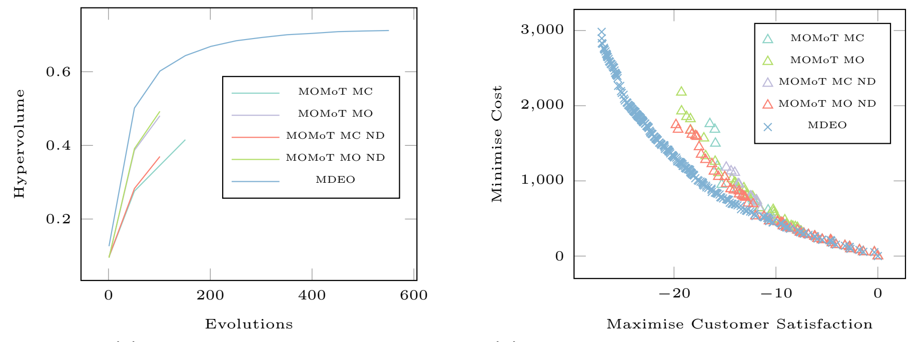
Despite this difference in the number of steps, for small models MOMoT is on par with or even better than MDEOptimiser in terms of solution quality (Table 4). In the CRA case, MOMoT finds a slightly higher objective median for input models A and B in the single-objective as well as in the multi-objective variants. The same can be observed for model A of the multi-objective REF case. However, as the size of the evaluated models increases, the quality of the solutions found by MOMoT decreases compared to solutions generated by MDEOptimiser. Most notably this can be seen for models C, D, and E of the CRA case and model B of the NRP case. In the single-objective variants this effect is also accompanied by a higher standard deviation.

	MDEO		MOMoT							
			MO		MO, ND		MC		MC, ND	
	MED	SD	MED	SD	MED	SD	MED	SD	MED	SD
CRA-A	2.33	0.45	**3.00**	0.47	2.67	0.67	**3.00**	0.49	**3.00**	0.72
CRA-B	1.82	0.53	2.16	0.50	1.83	0.46	**2.72**	0.61	1.50	0.64
CRA-C	**2.22**	0.54	-0.10	1.36	-1.52	1.74	-2.90	2.54	-5.71	3.49
CRA-D	**5.44**	0.88	-4.34	3.45	-5.82	3.12	-6.41	3.31	-15.45	5.29
CRA-E	**11.30**	0.95	-8.42	3.14	-10.04	3.97	-9.98	2.96	-20.13	4.83
CRA-EXT-A	0.89	0.04	**0.97**	0.05	0.90	0.07	**0.97**	0.06	**0.97**	0.06
CRA-EXT-B	0.89	0.02	**0.91**	0.02	0.89	0.03	0.90	0.03	0.88	0.03
CRA-EXT-C	**0.97**	0.02	0.94	0.04	0.93	0.03	0.86	0.06	0.87	0.06
CRA-EXT-D	**0.93**	0.03	0.81	0.05	0.87	0.04	0.71	0.06	0.76	0.05
CRA-EXT-E	0.90	0.04	0.82	0.05	**0.91**	0.03	0.71	0.08	0.81	0.04
NRP-A	**0.79**	0.00	0.76	0.02	0.77	0.02	0.75	0.03	0.75	0.02
NRP-B	**0.71**	0.01	0.48	0.04	0.49	0.04	0.42	0.05	0.38	0.06
REF-A	28.00	0.00	28.00	0.00	-	-	28.00	0.00	-	-
REF-B	51.60	0.00	51.60	0.00	-	-	51.60	0.00	-	-
REF-EXT-A	0.46	0.00	**0.53**	0.00	-	-	**0.53**	0.00	-	-
REF-EXT-B	0.50	0.00	0.50	0.00	-	-	0.50	0.00	-	-

Table 4 – Median results (MED) and standard deviations (SD) over 30 runs. The median objective value is shown for the single-objective, the median Hypervolume for the dual-objective variants. Generally, higher values are better. Only for single-objective REF, lower values are better. ND variants are not available for the REF case.

Configurations	PFS	PFC	BSR	Configurations	PFS	PFC	BSR
MOMoT MC CRA A	1	1	1	MOMoT MO CRA E	54	0	0
MOMoT MO CRA A	1	1	1	MOMoT MC ND CRA E	54	0	0
MOMoT MC ND CRA A	1	1	1	MOMoT MO ND CRA E	54	1	0.019
MOMoT MO ND CRA A	1	1	1	MDEO CRA E	54	53	0.981
MDEO CRA A	1	0	0	MOMoT MC NRP A	32	24	0.75
MOMoT MC CRA B	4	0	0	MOMoT MO NRP A	32	28	0.875
MOMoT MO CRA B	4	0	0	MOMoT MC ND NRP A	32	25	0.781
MOMoT MC ND CRA B	4	0	0	MOMoT MO ND NRP A	32	30	0.938
MOMoT MO ND CRA B	4	1	0.25	MDEO NRP A	32	31	0.969
MDEO CRA B	4	3	0.75	MOMoT MC NRP B	245	29	0.118
MOMoT MC CRA C	10	0	0	MOMoT MO NRP B	245	19	0.078
MOMoT MO CRA C	10	0	0	MOMoT MC ND NRP B	245	28	0.114
MOMoT MC ND CRA C	10	0	0	MOMoT MO ND NRP B	245	30	0.122
MOMoT MO ND CRA C	10	1	0.1	MDEO NRP B	245	245	1
MDEO CRA C	10	9	0.9	MOMoT MC Ref A	7	7	1
MOMoT MC CRA D	44	0	0	MOMoT MO Ref A	7	7	1
MOMoT MO CRA D	44	0	0	MDEO Ref A	7	5	0.714
MOMoT MC ND CRA D	44	0	0	MOMoT MC Ref B	24	24	1
MOMoT MO ND CRA D	44	1	0.023	MOMoT MO Ref B	24	24	1
MDEO CRA D	44	43	0.977	MDEO Ref B	24	22	0.917
MOMoT MC CRA E	54	0	0				

Table 5 – Summary of the PF_{pseudo} Size (PFS), number of PF_{pseudo} Contributions (PFC) and Ratios of Best Solutions found (BSR) for MDEO and MOMoT in all configurations.



(a) Median HV growth

(b) Pareto front approximations after timeouts were reached for each evaluated configuration

Figure 6 – Summary of the median HV increase over the number of evolution steps and Pareto front approximations for all evaluated configurations for model B of the NRP.

The quality of the solutions is confirmed by the reference set contributions observed for the multi-objective configurations included in Table 5. The table shows the total size of the PF_{pseudo} found for each problem instance, and the BSR rate, indicating the percentage of the PF_{pseudo} solutions found by each configuration.

Impact of Rule-based Crossover (Q2) While applying crossover affects the runtime, it does not do so in a consistent way ([JBB+]). In the CRA case, fewer steps could be performed when crossover was used. This effect is stronger for smaller models where mutation-only allows up to 50% more steps to be executed. For the largest models a difference of 15-20% can still be observed and the effect is a bit stronger for multi-objective variants than for single-objective ones. In contrast, applying crossover allows for up to 50% more steps in the NRP case. In the Refactoring case, there is no clear trend.

As shown in Table 4, in terms of solution quality, crossover is not beneficial for any but a single configuration (model B of the single-objective CRA case). Even in the NRP case, when more steps are executed with crossover, the results are worse.

Impact of Destructive Rules (Q3) The rejection of destructive rules has contrasting effects. Although the optimization is faster in the single-objective CRA case, solution quality decreases considerably. For the larger models of the multi-objective CRA version, however, we find the opposite to be true. In the NRP case, the rule execution speed decreases but the quality is barely affected (Table 4 and [JBB+]).

General Findings In general, the convergence rate differs between MDEOptimiser and MOMoT. MDEOptimiser needs fewer evolution steps to develop good solutions in all cases studied. Regarding the BSR, MDEOptimiser outperforms MOMoT clearly for the multi-objective CRA cases and model B of the NRP (Table 5). Additionally, the solutions of found Pareto front approximations spread wider across the search space, a condition usually considered favorable in multi-objective optimization [ČLM13]. Figure 6 shows the convergence behavior and the spread of the Pareto front approximations on the example of model B of the NRP case. For the other cases we refer the reader to the online archive [JBB+].

6 Discussion

As the rule-based approach allows more options (crossover, reconsidering past decisions, specialized rule sets) in supporting a search, one might expect it to dominate the model-based approach in at least one of the tested configurations. However, the evaluation results paint a different picture. In the following, we will discuss possible reasons for these observations along the associated research questions.

Preface As discussed in the qualitative comparison, in the rule-based approach rule calls may become invalid when changes are applied to their containing sequence. We find this to be a central aspect for the explanation of observations regarding runtime and solution quality. Due to the repair strategy used in the experiments, as rule calls become invalid in an evolution step the number of rule calls in a solution decreases. This positively affects runtime as fewer rules need to be applied when evaluating the fitness of that solution. We refer to this as *Invalidation Runtime Effect (IRE)* in the following. On the other hand, important evolutionary information might get lost and quality might degrade. We call this the *Invalidation Quality Effect (IQE).*

With an increasing size of sequences invalidations become more likely. As such, both effects become stronger for larger models where longer solution sequences are needed.

Comparing Approaches (Q1) The rule-based approach suffers from a slower evolution compared to that of the model-based approach. This is mainly caused by the overhead in applying all rule calls stored in a solution before the solution's fitness can be calculated. The overhead becomes more prominent for larger models where longer solution sequences are needed. This behavior reflects in the solution quality. MOMoT outperforms MDEOptimiser for most of the smaller models because it seems to have enough time to properly converge. As models get larger, aggravated by a stronger IQE, MOMoT is not able to converge equally well. A relatively high standard deviation in these cases underpins this theory.

Impact of Rule-based Crossover (Q2) In most of the cases, crossover was detrimental to the solution quality obtained by the rule-based approach. We attribute this effect to the destructive nature of the traditional crossover. By possibly invalidating many rule calls at once when mixing up sequences of rule calls, a high IQE kicks in. In the multi-objective REF case and the NRP case, applying crossover allowed to perform a higher number of evolution steps. Two possible reasons come to mind. Because the rules of these cases are potentially changing a large number of model elements at once, the invalidation of a single rule call might cause a snowball effect. Accordingly, the IRE might be more visible here than in the other use cases. Additionally, the IQE might degrade the quality of solutions so frequently that only a very small set of Pareto optimal solutions needs to be maintained. A faster selection process might be the result.

Impact of Destructive Rules (Q3) In the CRA and NRP cases, any optimal solution would certainly not need destructive rules. However, without them, solutions might have to be degraded first (e.g. by substituting a feature assignment with a class creation in the CRA case) before a better solution can be reached. Here, destructive rules might help in overcoming local optima. Unfortunately, this does not explain why the absence of such rules is beneficial for some of the use cases. More research is needed to discover the causalities of the observed behavior.

General Findings Generally, the rule-based approach converges slower than the model-based approach, i.e., more evolution steps are needed to reach a comparable solution quality. We attribute this to the IQE which may introduce steps of quality regression into the optimization process. As discussed for Q2, crossover adds to that problem in most cases.

7 Threats To Validity

Regarding research question Q1, the validity of our results depends highly on whether the observed differences in performance can be attributed to the differences in encoding. To mitigate the risk of side effects caused by the implementation of both encoding approaches we have chosen frameworks which are built on a similar basis. Both are implemented in Java, rely on EMF and Henshin for the modelling part, and use the same NSGA-II implementation for running the optimization. For common parameters, we used the same values and the same transformation rules were used to explore the search space. Where necessary, we also aligned the implementation of the tools: both starting from the same initial population in our study and performing mutations of equal size. However, although we checked key parts of both implementations, differences influencing the runtime cannot be ruled out completely.

The generalizability of our findings is limited as both approaches may expose different optimization behavior when other configurations, search operators and problems are selected. We are confident, though, that the characteristics of our use cases cover a wide range of problems of practical importance.

8 Related Work

Our work is related to various approaches to encode search problems, including conventional ones such as binary and integer representations. We discuss relevant work below.

8.1 Conventional encodings

Several types of encodings have been proposed for finding optimal genotype representations in evolutionary computation [ES15]. *Binary representation* uses a bit-string in which values of bits are interchanged and switched by the search operators to change the control variables. *Integer representation* and *real-valued representation* are similar to the previous category, however, represent the control variables as integers and real values, respectively. *Tree representation* uses a tree to represent the genotype. This encoding is common to represent syntax trees of programs, when using genetic algorithms to improve programs. *Graph representations* use a graph to represent both the genotype and the phenotype.

In many cases, there is an additional genotype-phenotype translation step required to convert an encoding to the solution candidate, such that it can be evaluated by the fitness functions. Usually, an additional repair step is needed after the application of search operators, in order to repair syntactically correct solutions which are semantically invalid.

8.2 Encodings in Model-Driven Optimization

Model-based encodings have been introduced by Burton et. al [BPR$^+$12], who used models and transformations to encode and evolve solution candidates for the NRP problem. The Crepé Complete framework is a more general approach that employs models to represent solutions for any given search problem [EWZ14]. However, this approach converts the models to an integer representation. FitnessStudio [Str17] uses the model-based encoding to generate efficient mutation operators. It evolves the mutation operators on a given training model using higher-order transformations.

8.3 Ruled-based encoding

The rule-based encoding has been introduced by Kessentini et. al [KLW13], who propose formulating search problems as a search of optimal transformation chains resulting from rule call sequences. Abdeen et al. [AVS$^+$14] follow this idea in their VIATRA-DSE framework, calling their approach *rule-based design space exploration*. Whereas VIATRA-DSE employs the VIATRA model transformation language to specify transformations, our comparison focuses on the MOMoT [BFT$^+$17] and MDEOptimiser [BZS18] tools, since they both use Henshin as the underlying transformation language.

8.4 Comparisons between encodings

Previous efforts to compare different encodings have focused on conventional encodings. Janikov et al. [JM91] experimentally compare a binary and a floating point encoding of a dynamic control problem. They find that some benefits of the binary encoding can be countered with refined problem-specific operators for the floating point encoding. Wu and Lindsay [WL96] compare two different flavors of linear representation, one with fixed and one with floating locations for the building blocks for individuals. They find that the floating representation enables the algorithm to better maintain diversity of individuals, and suggest to use both encodings in combination. Kantschik and Banzhaf [KB01] compare a text-based, a linear, and a hybrid representation and show that the hybrid outperforms the former two in most cases. To our knowledge, no work has compared different encodings for SBMDE problems, yet.

9 Conclusion

In this paper, we performed a qualitative and quantitative comparison of the two main encoding approaches in model-driven optimization, the model-based and the rule-based one. The quantitative comparison showed that the model-based approach tends to be more effective, except for the smallest considered models. While the ability of the rule-based approach to reconsider past decisions may be beneficial on the long run, it is quite likely the cause of a slower convergence compared to the model-based approach.

Interestingly, the main distinguishing features of the rule-based encoding, do not help very much. While the rejection of destructive rules caused a slight improvement in some problem cases, it is not a game changing factor in general. For traditional crossover, the result is even worse as it had a detrimental effect on solution quality most of the time. Typically, crossover can help if there are good parts in solutions which can be recombined. This can be problematic in the case of model transformations, as the dependency of rule calls need to be considered when multiple rule calls are exchanged during recombination. A crossover operator more tailored to the needs of MDO is needed here.

Finally, our analysis raised a couple of questions left for future work. How do the specific characteristics of a use case influence the performance of each encoding approach? How can the traditional crossover be improved to be more effective in the rule-based approach? And can crossover be done effectively in the model-based approach?

References

[ABJ+10] Thorsten Arendt, Enrico Biermann, Stefan Jurack, Christian Krause, and Gabriele Taentzer. Henshin: Advanced concepts and tools for in-place EMF model transformations. In *Int. Conference on Model Driven Engineering Languages and Systems*, pages 121–135, 2010. doi:10.1007/978-3-642-16145-2_9.

[AVS+14] Hani Abdeen, Dániel Varró, Houari Sahraoui, András Szabolcs Nagy, Csaba Debreceni, Ábel Hegedüs, and Ákos Horváth. Multi-objective optimization in rule-based design space exploration. In *Int. Conference on Automated Software Engineering*, pages 289–300, 2014. doi:10.1145/2642937.2643005.

[BBL10] M. Bowman, L. C. Briand, and Y. Labiche. Solving the Class Responsibility Assignment problem in object-oriented analysis with multi-objective genetic algorithms. *IEEE Transactions on Software Engineering*, 36(6):817–837, 2010. doi:10.1109/TSE.2010.70.

[BDW98] Lionel C. Briand, John W. Daly, and Jürgen Wüst. A unified framework for cohesion measurement in object-oriented systems. *Empirical Software Engineering*, 3(1):65–117, 1998. doi:10.1023/A:1009783721306.

[BFT+17] Robert Bill, Martin Fleck, Javier Troya, Tanja Mayerhofer, and Manuel Wimmer. A local and global tour on MOMoT. *Software & Systems Modeling*, pages 1–30, 2017. doi:10.1007/s10270-017-0644-3.

[BP13] Frank R Burton and Simon Poulding. Complementing metaheuristic search with higher abstraction techniques. In *Int. Workshop on Combining Modelling and Search-Based Software Engineering*, pages 45–48, 2013.

[BPR+12] Frank R Burton, Richard F Paige, Louis M Rose, Dimitrios S Kolovos, Simon Poulding, and Simon Smith. Solving acquisition problems using model-driven engineering. In *European Conference on Modelling Foundations and Applications*, pages 428–443, 2012. doi:10.1007/978-3-642-31491-9_32.

[BSAN17] Ilhem Boussaïd, Patrick Siarry, and Mohamed Ahmed-Nacer. A survey on search-based model-driven engineering. *Automated Software Engineering*, 24(2):233–294, 2017. doi:10.1007/s10515-017-0215-4.

[BZ16] Alexandru Burdusel and Steffen Zschaler. Model optimisation for feature class allocation using MDEOptimiser: A TTC 2016 submission. In *Transformation Tool Contest*, pages 33–38, 2016.

[BZ18] Alexandru Burdusel and Steffen Zschaler. Towards automatic generation of evolution rules for model-driven optimisation. In *Int. Workshop on Graph Computation Models*, pages 60–75, 2018. doi:10.1007/978-3-319-74730-9_6.

[BZS18] Alexandru Burdusel, Steffen Zschaler, and Daniel Strüber. MDEOptimiser: A search based model engineering tool. In *Int. Conference on Model Driven Engineering Languages and Systems*, pages 12–16, 2018. doi:10.1145/3270112.3270130.

[CCRS04] Carlos A. Coello Coello and Margarita Reyes Sierra. A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm. In *Advances in Artificial Intelligence*, pages 688–697, 2004. doi:10.1007/978-3-540-24694-7_71.

[ČLM13] Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Comput. Surv.*, 45(3):35:1–35:33, 2013. doi:10.1145/2480741.2480752.

[DPAM02] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. doi:10.1109/4235.996017.

[DZA+11] Juan J Durillo, Yuanyuan Zhang, Enrique Alba, Mark Harman, and Antonio J Nebro. A study of the bi-objective next release problem. *Empirical Software Engineering*, 16(1):29–60, 2011. doi:10.1007/s10664-010-9147-3.

[ES15] AE Eiben and JE Smith. *Introduction to evolutionary computing.* Springer, 2015.

[EWZ14] Dionysios Efstathiou, James R. Williams, and Steffen Zschaler. Crepe complete: Multi-objective optimisation for your models. In *Int. Workshop on Combining Modelling with Search- and Example-Based Approaches*, 2014.

[FSK17] S. Faridmoayer, M. Sharbaf, and S. Kolahdouz-Rahimi. Optimization of model transformation output using genetic algorithm. In *Int. Conference on Knowledge-Based Engineering and Innovation*, pages 0203–0209, 2017. doi:10.1109/KBEI.2017.8324973.

[FTCW16] Martin Fleck, Javier Troya Castilla, and Manuel Wimmer. The class responsibility assignment case. *Transformation Tool Contest*, pages 1–8, 2016.

[FTK+17] M. Fleck, J. Troya, M. Kessentini, M. Wimmer, and B. Alkhazi. Model transformation modularization as a many-objective optimization problem. *IEEE Transactions on Software Engineering*, 43(11):1009–1032, 2017. doi:10.1109/TSE.2017.2654255.

[GJ79] Michael R Garey and David S Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., 1979.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning.* Addison-Wesley, 1989.

[HJ98] Michael Pilegaard Hansen and Andrzej Jaszkiewicz. Evaluating the quality of approximations to the non-dominated set. Technical report, 1998.

[HJ01] Mark Harman and Bryan F Jones. Search-based software engineering. *Information and Software Technology*, 43(14):833–839, 2001.

[HMZ12] Mark Harman, S. Afshin Mansouri, and Yuanyuan Zhang. Search-based software engineering: Trends, techniques and applications. *ACM Comput. Surv.*, 45(1):11:1–11:61, 2012. doi:10.1145/2379776.2379787.

[JBB+] Stefan John, Alexandru Burdusel, Robert Bill, Daniel Strüber, Gabriele Taentzer, Steffen Zschaler, and Manuel Wimmer. Searching for optimal models: Comparing two encoding approaches - accompanying data. http://dx.doi.org/10.6084/m9.figshare.8236505. doi:10.6084/m9.figshare.8236505.

[JM91] Cezary Z Janikow and Zbigniew Michalewicz. An experimental comparison of binary and floating point representations in genetic algorithms. In *Int. Conference on Genetic Algorithms*, pages 31–36, 1991.

[JOZF14] S. Jiang, Y. Ong, J. Zhang, and L. Feng. Consistencies and contradictions of performance metrics in multiobjective optimization. *IEEE Transactions on Cybernetics*, 44(12):2391–2404, 2014. doi:10.1109/TCYB.2014.2307319.

[KB01] Wolfgang Kantschik and Wolfgang Banzhaf. Linear-tree GP and its comparison with other GP structures. In *European Conference on Genetic Programming*, pages 302–312, 2001. doi:10.1007/3-540-45355-5_24.

[KLW13] Marouane Kessentini, Philip Langer, and Manuel Wimmer. Searching models, modeling search: On the synergies of SBSE and MDE. In *Int. Workshop on Combining Modelling and Search-Based Software Engineering*, pages 51–54, 2013. doi:10.1109/CMSBSE.2013.6604438.

[MOE] MOEA framework. moeaframework.org/ (Accessed June 10, 2019).

[RKL+18] Sebastian Ruland, Géza Kulcsár, Erhan Leblebici, Sven Peldszus, and Malte Lochau. Controlling the attack surface of object-oriented refactorings. In *Int. Conference on Fundamental Approaches to Software Engineering*, pages 38–55, 2018. doi:10.1007/978-3-319-89363-1_3.

[RLB15] N. Riquelme, C. Von Lücken, and B. Baran. Performance metrics in multi-objective optimization. In *Latin American Computing Conference*, pages 1–11, 2015. doi:10.1109/CLEI.2015.7360024.

[SBG+17] Daniel Strüber, Kristopher Born, Kanwal Daud Gill, Raffaela Groner, Timo Kehrer, Manuel Ohrndorf, and Matthias Tichy. Henshin: A usability-focused framework for EMF model transformation development. In *Int. Conference on Graph Transformation*, pages 196–208, 2017. doi:10.1007/978-3-319-61470-0_12.

[SBMP08] Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. *EMF: Eclipse Modeling Framework*. Addison-Wesley, 2008.

[SNV18] Oszkár Semeráth, András Szabolcs Nagy, and Dániel Varró. A graph solver for the automated generation of consistent domain-specific models. In *Int. Conference on Software Engineering*, pages 969–980, 2018. doi:10.1145/3180155.3180186.

[Str17] Daniel Strüber. Generating efficient mutation operators for search-based model-driven engineering. In *Int. Conference on Theory and Practice of Model Transformations*, pages 121–137, 2017. doi:10.1007/978-3-319-61473-1_9.

[WL96] Annie S Wu and Robert K Lindsay. A comparison of the fixed and floating building block representation in the genetic algorithm. *Evolutionary Computation*, 4(2):169–193, 1996. doi:10.1162/evco.1996.4.2.169.

[ZBT07] Eckart Zitzler, Dimo Brockhoff, and Lothar Thiele. The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration. In *Evolutionary multi-criterion optimization*, pages 862–876, 2007. doi:10.1007/978-3-540-70928-2_64.

[ZM16] Steffen Zschaler and Lawrence Mandow. Towards model-based optimisation: Using domain knowledge explicitly. In *Workshop on Model-Driven Engineering, Logic and Optimization*, pages 317–329, 2016. doi:10.1007/978-3-319-50230-4_24.

About the authors

Stefan John is a PhD student at the Phillips-Universität Marburg. His current research interests are in model-driven engineering and optimization heuristics. Contact him at stefan.john@uni-marburg.de.

Alexandru Burdusel is a PhD student at King's College London. His research interests are in optimisation methods and model-driven engineering. Contact him at alexandru.burdusel@kcl.ac.uk

Robert Bill is a PhD student at TU Wien currently working as researcher for the Austrian Center of Digital Production. His research interest is in the broad field of model-driven engineering with a special focus on language engineering, model integration, and (model) optimization. Contact him at bill@big.tuwien.ac.at

Daniel Strüber is a post-doc at the software engineering division at Chalmers University at Technology and University of Gothenburg. His research interests are in model-driven engineering, search-based software engineering and software security. Contact him at danstru@chalmers.se, or visit http://www.danielstrueber.de/.

Gabriele Taentzer is professor in software engineering at the Philipps-Universität Marburg. Her research interests include formal foundations and applications of model-driven software engineering, graph transformation, and software quality assurance. Contact her at taentzer@informatik.uni-marburg.de, or visit http://www.uni-marburg.de/fb12/swt.

Steffen Zschaler is a senior lecturer in software engineering at King's College London. His research interests are in model-driven engineering, where he works on formal foundations, tooling (especially for search-based approaches to MDE and for reuse and composition of MDE artefacts), and applications. Contact him at szschaler@acm.org or http://www.steffen-zschaler.de.

Manuel Wimmer is a full professor leading the Institute of Business Informatics - Software Engineering at the Johannes Kepler University Linz. His current research interests are focused on the foundations and applications of model-driven engineering technologies. Contact him at manuel.wimmer@jku.at, or visit https://www.se.jku.at/manuel-wimmer.

Acknowledgments This work has been partially supported and funded by the Austrian Research Promotion Agency (FFG) via the *Austrian Competence Center for Digital Production* (CDP) under the contract number 854187, by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development, by the FWF under the grant numbers P28519-N31 and P30525-N31, and by the *Engineering and Physical Sciences Research Council* (EPSRC) under grant number 1805606.