Injecting Social Diversity in Multi-Objective Genetic Programming: the Case of Model Well-formedness Rule Learning

Edouard Batot and Houari Sahraoui

GEODES, DIRO, Université de Montréal
{batotedo, sahraouh}@iro.umontreal.ca

Abstract. Software modelling activities typically involve a tedious and time-consuming effort by specially trained personnel. This lack of automation hampers the adoption of the Model Driven Engineering (MDE) paradigm. Nevertheless, in the recent years, much research work has been dedicated to learn MDE artifacts instead of writing them manually. In this context, mono- and multi-objective Genetic Programming (GP) has proven being an efficient and reliable method to derive automation knowledge by using, as training data, a set of input/out examples representing the expected behavior of an artifact. Generally, the conformance to the training example set is the main objective to lead the search for a solution. Yet, single fitness peak, or local optima deadlock, one of the major drawbacks of GP, happens when adapted to MDE and hinder the results of the learning. We aim at showing in this paper that an improvement in populations' social diversity carried out during the evolutionary computation will lead to more efficient search, rapid convergence, and more generalizable results. We ascertain improvements are due to our changes on the search strategy with an empirical evaluation featuring the case of learning well-formedness rules in MDE with a multi-objective genetic algorithm. The obtained results are striking, and show that semantic diversity allows a rapid convergence toward the near-optimal solutions. Moreover, when the semantic diversity is used as the crowding distance, this convergence is uniform through a hundred of runs.

1 Introduction

Model Driven Engineering (MDE) aims at raising the level of abstraction of programming languages. MDE advocates the use of models as first-class artifacts. It combines domain-specific modeling languages to capture specific aspects of the solution, and transformation engines and generators in order to move back and forth between models while ensuring their coherence, or to produce from these models low level artifacts such as source code, documentation, and test suites [1]. Still, designing and developing artifacts able to perform automated tasks in MDE (ensuring the well-formedness of models, transforming models, etc.) requires one to have both knowledge in the targeted domain as well as in the design and development tools. If done manually, these activities typically involve a tedious and time-consuming effort by specially trained personnel. Such a lack of automation is considered by many MDE specialists as a threat to MDE adoption [2,3].

Yet, in recent years, many research contributions have shown that it is feasible to automatically learn how to perform a task through examples, or by analogy to similar, previously-solved tasks. More precisely, many of the proposed learning methods are based on Genetic Programming (GP) algorithms, and thereby promise to ease the burden of hand-programming growing volumes of increasingly complex information. As a matter of fact, empirical studies have shown a strong potential in learning automatically model transformations [4,5,6] and model well-formedness rules [7,8] from examples of tasks input/outputs. An *example* here must be understood as a couple <*input model; expected output*> defining the constraints that bind artifacts' output to input. The set of training examples represents the expected behavior of the artifact to learn and thus constitutes a convenient objective to lead the search of a solution.

Genetic programming and more generally multi-objective evolutionary computation has received increasing attention in the last decades. From early works [9,10,11,12], authors have formulated the idea that optimizing for multi-objective is to search for multiple solutions, each of which satisfy the different objectives to different degrees. The selection of the final solution with a particular combination of objectives' values is thus postponed until a time when it is known what combinations exist [13]. Studies have shown the value of such technics and their suitability to real problems. However, from the very beginning, authors pointed out two major drawbacks to the application of genetic programming (GP): (i) diversity of populations is difficult to maintain during evolution, and populations tend to gather around a *single fitness peak*; and, (ii) individuals tend to grow unnecessarily in size – also called *bloating* effect.

Both bloating and single fitness peak symptoms have been well investigated by researchers since early works, and valuable research directions were proposed [14,15,16]. Nevertheless, while adapting GP as an automatic process to learn well-formedness rules from examples, we encountered these same scenarios in a great amount of runs. Solutions agree on finding the correct outputs for a large number of examples, but fail all on a few same examples – a *single fitness peak* is reached. The approach seems to favor solutions with a high fitness, i.e., a high percentage of correct output found, at the expense of the diversity of the solutions.

On promoting diversity, Vanneshi et al. showed in their work the superior importance of research on indirect semantic methods that *"act on the syntax of the individuals and rely on survival criteria to indirectly promote a semantic behavior"* [17]. Inasmuch as semantics are considered in GP as a vector of examples, MDE learning from examples methodology offers an auspicious support for such investigations.In the present study, we introduce a new Social Semantic Diversity Measure of individuals (inspired from Natural Language Processing) operating indirectly during the execution of a well-established multi-objective genetic algorithm [18]. We illustrate our work and assess its value in an empirical study featuring the problem of automatic learning of well-formedness rules from examples and counter examples.

The following section draws a map of the two main drawbacks of genetic programming and how researchers tackle them. Section 3 details how employing our Social Semantic Diversity Measure foster efficiency and accuracy of a GP run. We illustrate our approach in a case study depicted in Section 4. We assess our assumption through an empirical evaluation in Section 5 Section 6 concludes briefly.

2 Background, Related Work, and Problem Statement

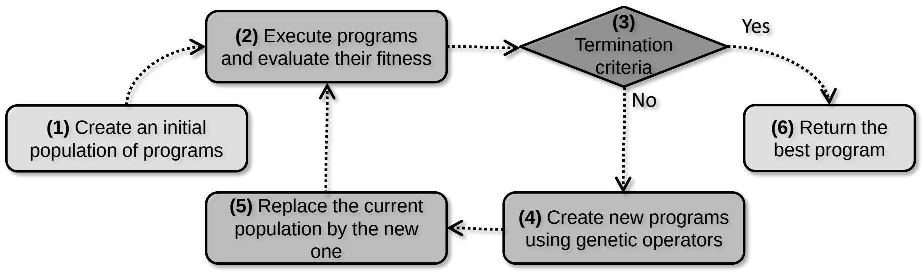

(1) Create an initial population of programs
(2) Execute programs and evaluate their fitness
(3) Termination criteria
Yes
No
(4) Create new programs using genetic operators
(5) Replace the current population by the new one
(6) Return the best program

Fig. 1: A typical genetic programming cycle

Genetic Programming (GP) execution is best understood using Fig. 1. At the beginning, an initial population of programs must be created (1). Then, every program of the population is executed the example inputs, and its fitness evaluated by comparing its outputs to the expected ones (2). If a termination criterion is reach (3), the solution program (or a set of near-optimal solutions, in case of multi-objective) is returned (6). Otherwise, a new population of programs is created by genetic operations (crossover and mutation) applied on selected potential reproducers (4). The new population replaces the previous one (5), and a new iteration starts (2). The loop is repeated until a termination criterion is reach (commonly, a perfect fitness, or an arbitrary large number of iterations). Although this process allows to find good solutions for many problems, it is known to suffer from two issues, *bloating* and *single fitness peak*. In the remainder of this section, we briefly discuss the *bloating* issue, and then focus more of the *single fitness peak* issue and its relation to diversity, which is the main object of this paper.

2.1 Bloating

Luke *et al.* suggest that, from a high level perspective, *bloating* (or *code growth*) happens because adding genetic material to individuals is *more* positively correlated to the fitness than removing material. They define it as the "uncontrolled growth of the average size of an individual in the population" [16]. Nonetheless, much work has been done to reduce the effect of bloating, offering to present readers a few options to choose from [15]. More precisely, in a multi-objective context, Pareto-based Multi-objective Parsimony Pressure (*i.e.,* using an objective devoted to constraining size of individuals) has been found very effective – with limited side effects [13,19]. We use this technique in our experiments.

2.2 Single fitness peak

The second problem with GP is the risk of a *single fitness peak* [13], consisting in a premature convergence together with a loss of diversity. Candidate solutions get stuck in a local optima and often no further improvement in fitness is noticed [20]. To tackle this issue, the level of diversity a population conveys must be given due consideration during a GP run [21]. More precisely, two phases of such a run are appropriate: at the initial population creation, to ensure a broad genetic material base; and/or during the evolution itself, to ensure that diversity does not fall from one generation to another. In both cases, diversity exists in two kinds: genotypic diversity considers the level of variability in individuals' structure, whereas phenotypic diversity focuses on the behavior of individuals.

Genotypic Diversity Genotypic diversity is the variety of individuals among a population with regards to their structure. It's a measure of the distance between individuals' syntax [22,23]. MDE though, since the syntax of artifacts is (very) complex, does not bare a single consensual definition of genotypic (or structural) diversity [24,25]. Nonetheless, to bestow a sufficiently diverse genetic material to start an evolutionary computation with, teams have used different metrics based on coverage estimations and showed interesting results. Works vary in nature and offer automatic generation of *diverse* models [8,26], or a user visual assistance helping when eliciting learning inputs data [27,28,29]. In any case, both techniques can be employed to provide with diverse initial population of solutions as well as with qualified input data.

Phenotypic Diversity As opposed to genotypic diversity, phenotypic diversity is measured on the behavior of a program – independently to its syntax. A phenotypic (or semantic [17]) measure, refers to the proportion of examples correctly processed by a program (*i.e.,* producing the expected output when executed on a specific input). It is a tangible fact that phenotypic diversity is more efficient than genotypic diversity to avoid the single fitness peak problem [17]. Nonetheless, if some early studies went as far as to expand the Darwinian metaphor and considered preference between individuals during GP run [30], to the best of our knowledge, there exists no study explicitly measuring benefits of phenotypic diversity when learning MDE artifacts.

Indirect Semantic Diversity Methods Roughly speaking, these methods combine both genotypic and phenotypic diversities. The rationale behind indirect diversity methods lies in their ability to distinguish between the aim of the method: individuals with acute Semantic Fitness, and the mean of its application: genetic modifications performed on their syntax. Understood as such, the heuristic remains agnostic of its mean of achievement and is ready to convey a strong generalization potential [31]. Vanneshi *et al.* [17] have proven the power of indirect diversity methods and call for more research in this field. It is to note here that, in the context of learning artifacts from examples in MDE, Semantic Fitness measure is a built-in feature and comes at no extra cost.

3 Social Semantic Diversity Measure

Nonetheless, MDE-artifacts learning from examples might be perfectly fit to GP adaptation, single fitness peaks yet keep happening during evolution. This leads to a disproportionate number of solutions with a good fitness, at the expense of their diversity.

Processing most examples correctly, these *alphas* [14] struggle to solve all examples exhaustively. Meanwhile, unfortunately, solutions able to solve the remaining corner cases reach a (much) lower fitness. Withal, since reproducers are chosen with regard to their fitness, the genetic material these latter *partial solutions* convey is lost and *corner cases* are never solved. A remedy to this deficiency was found using a social diversity measure.

We call *Social Diversity Measure* a measure that does not take into account the only individualistic fitness (*i.e.,* how many examples an individual resolves) but considers as well a social dimension (*i.e.,* what does that individual bring to the general fitness of the population).

Since we use a Semantic fitness, the remaining of this paper will mention Social Semantic Diversity Measure (SSDM). Its computation, based on the *inverse example resolution frequency* (IERF) is inspired from the *term frequency-inverse document frequency* (TF-IDF) numerical statistic [32] in the information retrieval domain. In other words, the SSDM of a solution is the sum of IERF of the examples it solves.

Paraphrasing *TFIDF* definition may help the reader to grasp the general idea of SSDM. We formulated as follows: "SSDM increases proportionally to the number of examples solved and is offset by the frequency of which an example is solved by the population's individuals, which helps to adjust for the fact that some examples are more frequently solved in general."

As a consequence, SSDM favors solutions solving *corner cases* by considering how many solutions in the population solve an example.

4 Learning Well-formedness Rule

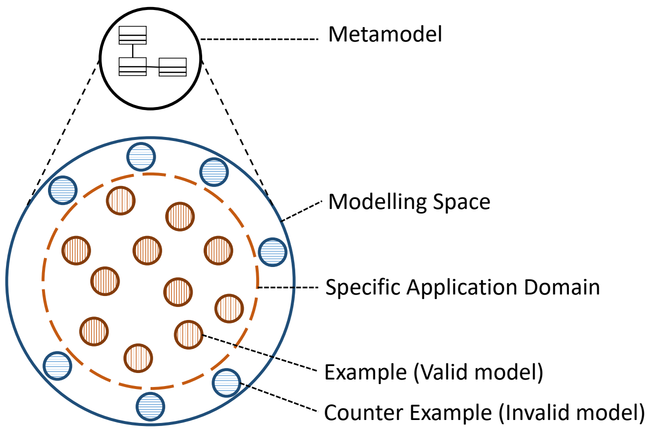


Fig. 2: Metamodel, modelling space and application domain

In this section, we illustrate how social semantic diversity can be implement in a multi-objective genetic-programming algorithm to learn model well-formedness rules

(WFRs) from examples. As mentioned in the introduction, researchers offer to use GP to learn some of MDE artifacts automatically as a substantial alternative to writing them manually. Indeed, we aim at showing in this paper that, during the process, which scalability remains at stake [33], an improvement in populations' social diversity will lead to more efficient search and more generalizable results. Thus far, the reader is asked to understand the little space left for implementation details.

After a brief overlook at the use and function of well-formedness rules, we will depict how much a tangible support GP, and more precisely multi-objective GP, offers to learn them automatically from examples and counter examples.

4.1 Well-formedness rules

In the MDE paradigm, due to their high level of abstraction, metamodels usually define too-large modelling spaces. They must be enriched with constraints, or rules, limiting the scope of their possible instantiations, *i.e.,* well-formed models in contrast to ill-formed models. Fig. 2 schematizes the concept of specific application domain: a metamodel defines a modelling space (within blue line) ; of which a specific application domain is a sub-space (within red dashed line). A set of WFRs allows to automatically differentiate between valid (well-formed) and invalid (ill-formed) models – it formally describes the limit of that targeted specific domain.

Representation. In the context of a GP learning process, a solution to our problem is thus a set of WFRs. More precisely, we represent a WFR as a tree which nodes are logical operators (*AND*, *OR*, *IMPLIES*, and *NOT*) and first-order quantifiers (*forAll* and *exists*), and which leaves are learning atomic blocks in the form of *OCL patterns* instances. Consequently, a solution is a tree with as root a vector whose elements are pointers to the individual WFR trees. Fig. 3 shows an example of a candidate (not necessarily valid) solution with 3 WFRs for the state-machine metamodel. The first and second rules constrain a *final* state to have respectively one incoming transition and no outgoing transition. The third rule requires that a pseudostate *choice* must have at least one incoming or outgoing transition. As for their execution, we implement WFRs in the *defacto* language Object Constraint Language (OCL[1]).

OCL patterns. The rationale behind OCL patterns is beyond the scope of this paper. These result from empirical studies carried out on more than 400 metamodels from industry and academe alike [34]. In a nutshell, OCL patterns should be understood here as a minimalistic set of templates which instantiation and composition allows to express all and every *useful* WFR.

Size concern. Since solutions must be legible by final user (within human reach), the size of constraints must be kept as small as possible.

4.2 GP Adaptation

Our goal is to find the minimal set (*i.e.,* size) of WFRs that best discriminates between the valid and invalid example models (*i.e.,* fitness). Size and fitness objectives being

[1] http://www.omg.org/spec/OCL/

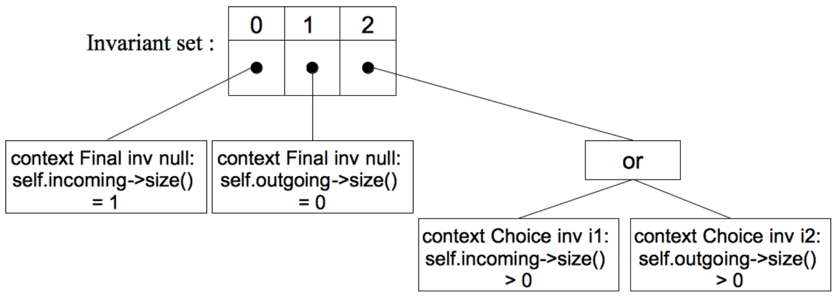

Invariant set :
0
1
2
context Final inv null: self.incoming->size() = 1
context Final inv null: self.outgoing->size() = 0
or
context Choice inv i1: self.incoming->size() > 0
context Choice inv i2: self.outgoing->size() > 0

Fig. 3: An example of solution containing 3 WFRs.

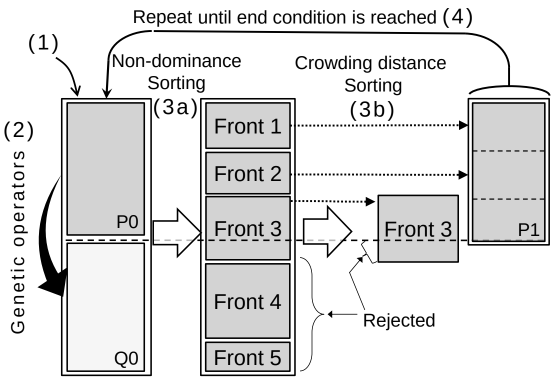

Repeat until end condition is reached (4)
(1)
Non-dominance Sorting (3a)
Crowding distance Sorting (3b)
(2)
Genetic operators
P0
Q0
Front 1
Front 2
Front 3
Front 4
Front 5
Front 3
P1
Rejected

Fig. 4: Non Sorting Genetic Algorithm NSGA-II [18]

different in nature, we represent the learning of WFRs as a multi-objective optimization problem, and we solve it using the Non-Sorting Genetic Algorithm NSGA-II [18].

The idea of NSGA-II [18] is to make a population of candidate solutions evolve toward the near-optimal solution in order to solve a multi-objective optimization problem. NSGA-II is designed to find a set of optimal solutions, called non-dominated solutions, also Pareto set. A non-dominated solution is the one which provides a suitable compromise between all objectives without degrading any of them. As described in Fig. 4, the first step in NSGA-II is to create randomly a population P_0 of $N/2$ individuals encoded using a specific representation (1). Then, a child population Q_0, of the same size, is generated from the population of parents P_0 using genetic operators such as crossover and mutation (2). Both populations are merged into an initial population R_0 of size N, which is sorted into dominance fronts according to the dominance principle (3a). A solution s_1 dominates a solution s_2 for a set of objectives $\{O_i\}$ if $\forall i, O_i(s_1) \geqslant O_i(s_2)$ and $\exists j \mid O_j(s_1) > O_j(s_2)$. The first (Pareto) front includes the non-dominated solutions; the second front contains the solutions that are dominated only by the solutions of the first front, and so on and so forth. The fronts are included in the parent population P_1 of the next generation following the dominance order until the size of $N/2$ is reached.

If this size coincides with part of a front, the solutions inside this front are sorted, to complete the population, according to a crowding distance which favors "diversity" in the solutions (3b). This process will be repeated until a stop criterion is reached, *e.g.,* a number of iterations or a certain value of the Semantic Fitness.

We adapted NSGA-II to our problem as follows.

– **Solution Representation and Creation.** A solution to our problem is represented as mentioned in Section 4.1, *i.e.,* a set of OCL constraints, each representing a WFR implemented as tree. The initial population is created randomly. For each individual, the average number of nodes in the WFR trees, the maximum depth, and the maximum width are configurable;
– **Reproduction.** As genetic operators, we use a single-point crossover applied to the tree-root vector, and two kinds of mutations. First, a node from a WFR tree is chosen randomly. If it is a leaf, the pattern instance is either replaced with a new randomly created one or, if applicable, the pattern parameters are replaced randomly with applicable values. If the selected node is a logical operator, this is changed randomly.
– **Objectives.** We consider three objectives: *Size* is the number of leaves in the constraint tree, the smaller the better; *Semantic Fitness* is the number of examples processed accurately by an individual, to be maximized; and *Diversity* is SSDM, which can be represented either as an objective or a crowding distance, to be maximized as well.
– **Termination criteria.** Evolution stops if either a Semantic Fitness of 99%, or an arbitrary large number of iterations, is reach.

4.3 Social Semantic Diversity Implementation

We offer to employ the Social Semantic Diversity Measure (SSDM) in two different ways. The first is as an objective of its own, considered together with above-mentioned size and fitness (as promoted by Dejong *et al.* [13]). The other builds on peculiar limitation of NSGA-II [35] and acts as an alternative to the computation of a crowding distance. In both cases, SSDM computation remains the same. This is explained in Listing 1.1.

More specifically, implementing SSDM comes to adapting TF-IDF [32] using examples as documents and solutions as words. At a given iteration, SSDM is calculated from a binary matrix in which each cell represents the score of an individual against an example of the training set. The frequency of an example is the number of times it is solved by individuals. Finally, individual's SSDM value is the sum of *inverse example resolution frequencies* of examples that it processes accurately. More precisely, variables are:

– $example\_set$, the vector of training examples;
– $sol\_vs\_examples$, which contains the result of the comparison between output of individuals and output of the oracle when executed on $example\_set$;
– and fq_ex, which contains examples frequencies, recording how many solutions solve each example from $example\_set$;
– $ierfi$, the vector of *inverse example resolution frequencies* of training examples.

Listing 1.1: Excerpt for SSDM weights calculation.

```
\\ Compute frequencies of examples solved
for (int i = 0; i < sol_vs_ex.length; i++)
  for (int j = 0; j < sol_vs_ex[i].length; j++)
    fq_ex[j] += sol_vs_ex[i][j];

\\ Inverse document frequencies
for (int j = 0; j < fq.length; j++)
  ierfi[j] = Math.log10(D/fq_ex[j]);

\\ Weigthing
weight = 0;
for(int j = 0; j < example_set.length; j++)
  if(example_set[j].isAccurate())
    weight += ierfi[j];
```

5 Evaluation

To assess the improvement brought by the social semantic diversity in the search strategy, we conducted an empirical evaluation[2]. We formulate our research questions as follows:

- **RQ0**: Are our results a consequence of an efficient exploration of the search space, or are they due to the vast number of individuals we consider during the evolution?
- **RQ1**: Does the use of Social Semantic Diversity as an objective improves the search strategy, and, if so, how much?
- **RQ2**: Does the use of Social Semantic Diversity as an alternative crowding distance exhibit better efficiency and generalizabilty than SSDM as an objective?

5.1 Setting

In order to mitigate the influence of a metamodel specific structure on the learning process, we selected three metamodels (`FamilyTree`, `Statemachine`, and `Project Manager`) that demonstrate different levels of structure complexity, and that require diverse OCL WFR sets. We provided with oracle (*i.e.,* expected WFRs) manually. In more details, `FamilyTree` is the most simple case. Yet, it has been used as an illustrative example in various publications in the MDE research literature, such as [36]. `Statemachine` illustrates structural cardinality restrictions and define a common, widely used language. Finally, `Project Manager` is the most complex case and comes from [37].

[2] All data from the experiment is available at URL anonymized

Learning examples To provide with example sets of *quality* (*i.e.,* covering at best the modelling space, yet as small as can be), we used a model generator [8]. Size matters since every generated model example must be, in a real setting, tagged manually as *valid* or *invalid*. For the sake of experiment, we use the WFRs oracles to mimic the manual tagging. To run the experiment, we used two sets of examples for each metamodel. On the one hand, 20 models (10 valid, 10 invalid) were required for the learning (a *training set*). On the other hand, 100 models (50 valid, 50 invalid) were used to measure solutions' accuracy (*test bench*).

Configurations and variables Four configurations were considered to illustrate and answer our research questions (see Section 4.2 for implementation details). **RND** is a random exploration of the search space that takes the best among a given number of solutions randomly generated; **STD** is a standard run of NSGA-II [18] with two objectives, size and semantic fitness; **OBJ** is a run of NSGA-II with three objectives: size, semantic fitness and SSDM diversity; and **CD** is a run of NSGA-II with size and semantic fitness as objectives and a SSDM crowding distance.

We used two dependent variables to quantify experiment results: **#GEN**, the number of generation the evolutionary computation needed to find a solution. A score of 3000 means that there was no solution with perfect fit found during the search, and **ACC**, the proportion of examples from the test bench a solution process accurately.

Evaluation protocol For the NSGA-II parameters, we use a maximum number of iterations of 3000 and a population size of 30 solutions. Crossover and mutation probabilities are set to 0,9 and 0,3 respectively. In addition, solutions are created with between 5 to 15 WFRs with each WFR having a maximum depth of 3 and width of 15. We answer RQ0 with a comparison between the results given when using SSDM as an objective (OBJ) in the search strategy and those of a random exploration (RND). Since our strategy explores 3000*30 solutions, the random exploration explores randomly 90000 solutions as well and considers the best individual so created. We answer RQ1 with a comparison between the solutions obtained after an execution with (OBJ) and one without (STD) social semantic diversity objective. Finally, we answer RQ2 by comparing the configurations with social semantic diversity objective (OBJ) and with social semantic diversity crowding distance (CD). We ran each treatment 100 times to tackle GP indeterminism and guarantee statistical significance of the findings using the Mann-Whitney test.

5.2 Results and Analysis

RQ0 - Sanity Check As can be seen in Table 1, the RND configuration gives very poor results in comparison with an OBJ execution for two most complex metamodels (average accuracy of 0.5 vs 0.76 for `Project Manager`and 0.53 vs. 0.94 for`Statemachine`). The difference in both cases is statistically significant (p-value <0,001) and the effect size is large (Cohen's d > 5). For the small metamodel `FamilyTree`, although statistically significant, the difference and the effect size are small. ***We can conclude that solutions are significantly more generalizable when using OBJ configuration***.

Table 1: Statistical comparison of results between random search and our approach on three WFR learning scenarios.

	Average ACC Value		Mann Witney p-value	Effect Size Cohen's d
	RDN	OBJ		
`Project Manager`	0.5	0.76	<0.001	7.35
`Statemachine`	0.53	0.94	<0.001	5.38
`FamilyTree`	0.93	0.98	<0.001	0.74

RQ1 - Social Semantic Diversity Method, an improvement? Efficiency shows a significant improvement when SSDM is used, as can be seen in odd columns of Fig. 5. The number of generations required to find a solution when employing OBJ is a lot smaller than when employing STD. With `Project Manager` metamodel, an STD run hardly find solutions solving all training examples within 3000 generations, but OBJ do it in an average of 260 generations. More, solutions were found with significantly better accuracy than STD (respectively 0.76 against 0,69) and thus strengthen solutions' generalizability likewise. This success is also noticed, if of lesser magnitude, during executions on the `Statemachine` metamodel. Here, if solutions are found in both configuration, yet OBJ is significantly faster (with 782 generations, when STD requires more than 1782). As for the `FamilyTree` metamodel (not shown if the figure), solutions given by OBJ executions output a similar ACC (0.98) but significantly faster with 25 generations (resp. 76 with STD). ***We can conclude that injecting the social semantic diversity significantly improves the learning results***.

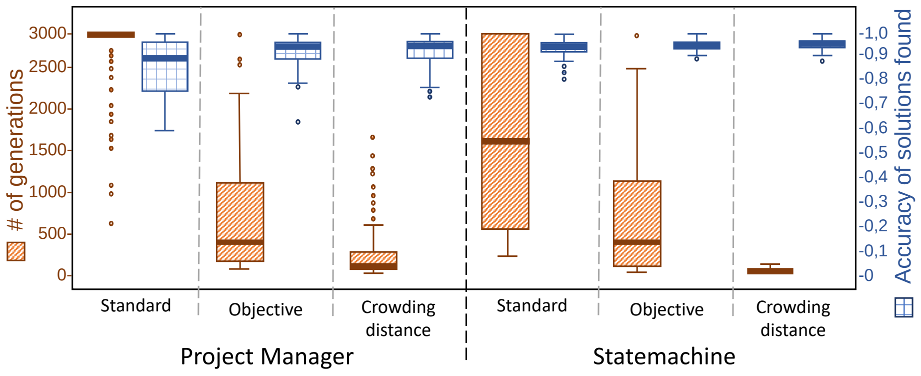
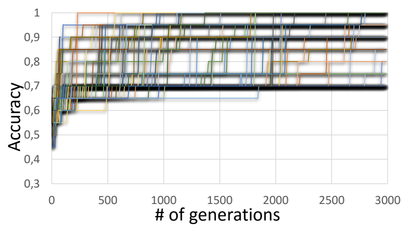
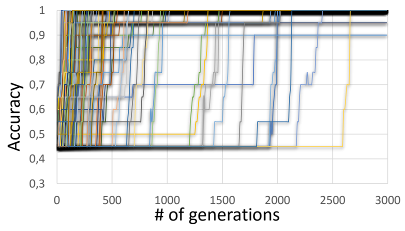


Fig. 5: Number of generations to find solutions and their accuracy on test bench for `Project Manager` and `Statemachine` metamodels.

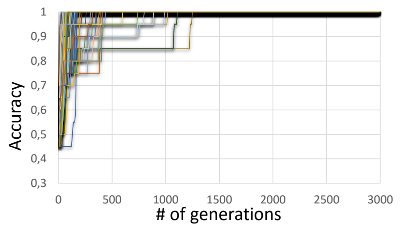
RQ2 - Social Semantic Diversity Method as an alternative crowding distance, any better yet? Results of RQ2 are flagrant (see the third configuration for both metamod-



(a) Standard evolution (STD)



(b) SSDM as an objective (OBJ)



(c) SSDM as crowding distance (CD)

Fig. 6: Evolution of individuals' average accuracy value during runs on `Project Manager` metamodel, with a hundred runs a plot.

els in Fig. 5). In Fig. 6, a hundred runs show together how using SSDM (6c and 6b) surges the learning curves and fosters solutions exploration, compared to a standard run (6a). As for generalizability, it doesn't seem that choosing between SSDM as an objective (OBJ) or in the crowding distance (CD) has any significant impact on the accuracy of solutions found (Mann Witney p-value > 0.01; see even columns in Fig. 5 for an illustration). Thence, the main difference lies in the smaller average number of iterations CD needs to converge, compared to OBJ runs. Note that that analysis is the strongest with `Project Manager` and `FamilyTree` metamodels. With `Statemachine` metamodel's results are slightly mitigated but remains significant. In that case, WFRs are more generally focused on structural cardinality than WFRs of the two other metamodels. We conceive this might be a factor for slightly different results. ***We can conclude that social semantic diversity as a crowding distance is more efficient than as an objective***.

In conclusion, as shown in Fig. 5 and Fig. 6 and certified with statistical analysis, the OBJ strategy surpasses significantly a STD exploration of solutions. Convergence is faster and output more generalizable (*i.e.,* confronting solutions to a test bench gives better results). A reason for these results might come from the way size is controlled. As recognized in the literature, we implemented it as a Pareto-based Multi-objective Parsimony Pressure. We noted, as expected [13], that solutions were skewed toward a 1.0 size, and the Pareto front grew large. Solutions size was indeed the one expected (*i.e.,* legible by a human), and the search, passed a few generations, relied mainly on Semantic Fitness. As a presumed consequence, when putting SSDM as an alternative to crowding distance, results were breathtaking on the three metamodels. Finally, using

Social Semantic Diversity Measure as an alternative crowding distance outperforms its use as an additional objective. Convergence is boosted, and generalizability is kept at its maximum. We hope these results are generalizable and claim the need to explore other applications, with OBJ and CD alike.

5.3 Thread to Validity

Although our approach produced good results on three metamodels, a threat to validity resides in the generalization of our approach to other scenarios. Still, metamodels show different characteristic and origin, and while our sample does not cover all learning scenarios, we believe that it is representative enough of a wide range of metamodels.

Another threat to the validity of our results relates to the use of a single set of (20) models to learn each WFR sets. Characterization of example sets is an ongoing investigation, and different sets might show different results. Yet, to mitigate what specificities the manual design of models can bring and encourage replication of our work, we used a generator [8]. Also, using the same set in every configuration ensures a difference in sets do not interfere in the experiment.

Regarding the applicability to other MDE artifacts, we believe that the idea to consider the social dimension of individuals' characteristics shall apply to the evolutionary computation of model transformation as well. In this case, *inverse example resolution frequency* could be used as well and we prospect, as future work, to replicate this study on model transformation learning.

Finally, we encourage further replication of our work to determine whether different multi-objective GP algorithms could benefit as well from our discovery.

6 Conclusion

This paper studies the impact of using a social semantic diversity to improve the search process for the multi-objective optimization problem of learning model well-formedness rules from examples and counter examples. The *Social Semantic Diversity* is measured (SSDM) in way that does not take into account the only individualistic fitness (*i.e.,* how many examples an individual resolves) but considers as well a social dimension *i.e.,* what does that individual bring to the general fitness of the population. We integrated SSDM in the NSGA-II algorithm as (i) an additional objective, and (ii) as an alternative to the crowding distance.

We evaluated the two options by learning WFRs for three metamodels. Our results are compiling evidence that injecting the social semantic diversity in the search process, especial as an alternative to the crowding distance, improves the convergence and the quality of the learned artifacts. The proposed measure and its integration in the multi-objective optimization algorithm are agnostic with respect to the learned artifact and the input/output examples used to guide the search. This allows to use social semantic diversity for a wide range of problem that can be solved by a multi-objective genetic programming algorithm. This claim must, however, be supported by replication studies. We expect to conduct some of these studies, especially, for model transformation learning.

References

1. D. C. Schmidt, “Model-driven engineering,” *IEEE Computer Society*, vol. 39, no. 2, 2006.
2. B. Selic, “What will it take? A view on adoption of model-based methods in practice,” *Int. J. on Soft. and Systems Modeling*, vol. 11, no. 4, pp. 513–526, 2012.
3. J. Whittle, J. Hutchinson, and M. Rouncefield, “The state of practice in model-driven engineering,” *Software, IEEE*, vol. 31, pp. 79–85, 2014.
4. M. Kessentini, W. Kessentini, H. Sahraoui, M. Boukadoum, and A. Ouni, “Design defects detection and correction by example,” in *Proc. of the Int. Conf. on Program Comprehension*, 2011, pp. 81–90.
5. H. Saada, X. Dolques, M. Huchard, C. Nebut, and H. A. Sahraoui, “Generation of operational transformation rules from examples of model transformations,” in *Proc. of the Int. Conf. on Model-Driven Engineering Languages and Systems*, 2012, pp. 546–561.
6. I. Baki and H. Sahraoui, “Multi-step learning and adaptive search for learning complex model transformations from examples,” *ACM Trans. on Soft. Eng. and Methodology*, vol. X, p. 36, 2015.
7. M. Faunes, J. Cadavid, B. Baudry, H. Sahraoui, and B. Combemale, “Automatically searching for metamodel well-formedness rules in examples and counter-examples,” in *Proc. of the Int. Conf. on Model-Driven Engineering Languages and Systems*, 2013, pp. 187–202.
8. E. Batot and H. Sahraoui, “A generic framework for model-set selection for the unification of testing and learning mde tasks,” in *Proc. of the Int. Conf. on Model-Driven Engineering Languages and Systems*. ACM, 2016.
9. J. D. Schaffer, “Multiple objective optimization with vector evaluated genetic algorithms,” in *Proceedings of the 1st International Conference on Genetic Algorithms*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1985, pp. 93–100.
10. D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.
11. J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992.
12. J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.
13. E. D. de Jong, R. A. Watson, and J. B. Pollack, “Reducing bloat and promoting diversity using multi-objective methods,” in *Proc. of the 3rd Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO’01, 2001, pp. 11–18.
14. T. F. Bersano-Begey, “Controlling exploration, diversity and escaping local optima in gp: Adapting weights of training sets to model resource consumption,” in *Late Breaking Papers at the 1997 Genetic Programming Conference*, J. R. Koza, Ed., 1997, pp. 7–10.
15. T. Soule and J. A. Foster, “Effects of code growth and parsimony pressure on populations in genetic programming,” *Evolutionary Computation*, vol. 6, no. 4, pp. 293–309, 1998.
16. S. Luke and L. Panait, “A comparison of bloat control methods for genetic programming,” *Evol. Comput.*, vol. 14, no. 3, pp. 309–344, September 2006.
17. L. Vanneschi, M. Castelli, and S. Silva, “A survey of semantic methods in genetic programming,” *Genetic Programming and Evolvable Machines*, vol. 15, no. 2, pp. 195–214, 2014.
18. K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, “A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: NSGA-II,” in *Int. Conf. on Parallel Problem Solving from Nature - PPSN*, 2000.
19. A. Ekárt and S. Z. Németh, “A metric for genetic programs and fitness sharing,” in *Genetic Programming*, R. Poli, W. Banzhaf, W. B. Langdon, J. Miller, P. Nordin, and T. C. Fogarty, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 259–270.

20. B. Wyns, P. De Bruyne, and L. Boullart, "Characterizing diversity in genetic programming," in *Proceedings of the 9th European Conference on Genetic Programming*, ser. EuroGP'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 250–259.
21. E. K. Burke, S. Gustafson, and G. Kendall, "Diversity in genetic programming: an analysis of measures and correlation with fitness," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 1, pp. 47–62, Feb 2004.
22. N. F. McPhee and N. J. Hopper, "Analysis of genetic diversity through population history," in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 2*. Morgan Kaufmann Publishers Inc., 1999, pp. 1112–1120.
23. N. F. McPhee, B. Ohs, and T. Hutchison, "Semantic building blocks in genetic programming," in *Genetic Programming*. Springer Berlin Heidelberg, 2008, pp. 134–145.
24. B. Baudry and M. Monperrus, "The multiple facets of software diversity: Recent developments in year 2000 and beyond," *ACM Comput. Surv.*, vol. 48, no. 1, pp. 16:1–16:26, 2015.
25. F. D. Giraldo, S. EspaÃśa, and O. Pastor, "Analysing the concept of quality in model-driven engineering literature: A systematic review," in *2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS)*, May 2014, pp. 1–12.
26. H. Wu, "Generating metamodel instances satisfying coverage criteria via smt solving," in *Proc. of the Int. Conf. on Model-Driven Eng. and Soft. Development*, 2016, pp. 40–51.
27. A. Ferdjoukh, F. Galinier, E. Bourreau, A. Chateau, and C. Nebut, "Measuring Differences To Compare Sets Of Models And Improve Diversity In MDE," in *ICSEA: International Conference on Software Engineering Advances*, Athenes, Greece, October 2017.
28. J. Sanchez-Cuadrado, J. de Lara, and E. Guerra, "Bottom-up meta-modelling: An interactive approach," in *Proc. of the Int. Conf. on Model-Driven Engineering Languages and Systems*, 2012, vol. 7590, pp. 3–19.
29. J. J. López-Fernández, E. Guerra, and J. de Lara, "Example-based validation of domain-specific visual languages," in *Proc. of the Int. Conf. on Software Language Engineering*, ser. SLE 2015, 2015, pp. 101–112.
30. C. Ryan, "Racial harmony in genetic algorithms," 1994.
31. V. K. Dabhi and S. Chaudhary, "A survey on techniques of improving generalization ability of genetic programming solutions," *CoRR*, vol. abs/1211.1119, 2012.
32. K. Sparck Jones, "Document retrieval systems," P. Willett, Ed., 1988, ch. A Statistical Interpretation of Term Specificity and Its Application in Retrieval, pp. 132–142.
33. M. Harman, Y. Jia, and Y. Zhang, "Achievements, open problems and challenges for search based software testing," in *Proc. of the Int. Conf. on Software Testing Verification and Validation*, 2015, pp. 1–12.
34. J. J. Cadavid, B. Combemale, and B. Baudry, "Ten years of Meta-Object Facility: an Analysis of Metamodeling Practices," AtlanMod, Research Report RR-7882, 2012.
35. F.-A. Fortin and M. Parizeau, "Revisiting the nsga-ii crowding-distance computation," in *Proc. of the 15th Conf. on Genetic and Evolutionary Computation*, ser. GECCO '13. ACM, 2013, pp. 623–630.
36. M. Gogolla, A. Vallecillo, L. Burgueno, and F. Hilken, "Employing classifying terms for testing model transformations," in *Proc. of the Int. Conf. on Model-Driven Engineering Languages and Systems*, 015, pp. 312–321.
37. K. Hassam, S. Sadou, and R. Fleurquin, "Adapting ocl constraints after a refactoring of their model using an mde process," in *9th ed. of the BElgian-NEtherlands software eVOLution seminar*, 2010, pp. 16–27.