Genetic Algorithm to Improve Diversity in MDE

Florian Galinier, Éric Bourreau, Annie Chateau, Adel Ferdjoukh, Clémentine Nebut

LIRMM, Université de Montpellier and CNRS, Montpellier, France
{lastname}@lirmm.fr

1 Introduction

Model Driven Engineering (MDE) provides tools for systematization and automation of models. Model transformations are nowadays widely used either in a maintenance context or in a more traditional development contexts. It allows realization at many stages during life cycle for a software: code generation, reverse engineering, etc. These models transformations are for example a way to change a model from a specific domain – *e.g.* a UML class model – to another domain – *e.g.* a Java model. Domains of models are specified by models of higher abstraction level: Meta-Models (MM).

Generation of models from MM as a testing purpose is a quite complex task. Several combinatorial approaches have already been proposed. In [1–3], the authors propose a generation based on Constraint Satisfaction Problems (CSP). These methods provide encouraging results, creating models that conform to meta-models and moreover, encoding constraints (OCL) that can be added to meta-models. Recently, in [4], simulated annealing generate a **set of models** which are the more dissimilar and which cover the meta-model. Meta-heuristics seem to provide interesting results in term of diversity, and, in our opinion, diversity is now a major issue to complete our generation process.

In this paper, we explain how to generate diverse models conform to a meta-model, using genetic algorithms.

2 Evolutionary algorithm

Evolutionary algorithm (EA) have been largely adopted to provide good solutions to combinatorial problems. In this section we will describe the differents components of our EA [5].

2.1 Chromosome representation

Representing models as a sequence in a chromosome is complex, moreover if we want to allow the possibility to crossover models. We base our approach on the CSP paradigm used by the authors in [1]. In this work, Ferdjoukh et al. propose to describe a model as a constraint network composed by variables, domains associated to these variables and a set of constraints. A valid model (*i.e.* which respect the meta-model and constraints) is an instantiation of variables that respect CSP constraints. We first choose to use vectors of instanciated values as chromosome representation. In a second time, we prefer to express in a chromosome a set of models (a list of size p) by concatenating p consecutive vectors representing p different models.

2.2 Initial Population

In [6], Ferdjoukh et al. introduce probabilistic simulation to generate realistic datas from a MM, based on a CSP instantiation. Their tool (named Grimm) is able to generate a set of valid models, conforms to meta-models, with variety in domain range instantiation due to probability distribution. A first set of 100 models, conforms to a specific meta model, is generated. Due to the value of p, we respectively considered 50 chromosomes (p=2), 25 chromosomes (p=4) or 20 (p=5).

2.3 Fitness function

Diversity between models is a rich problem. Indeed, it could be linked to a metrix representing similarity/dissimilarity. Some previous works in model driven engineering [7] used semantics data of models to provide a similarity measure. In our case, semantics data seems not sufficiently realistic, models are generated. Instead, we choose to focus on structure to define fitness.

As previously explained we reduced the problem by representing a graph as a vector including for each artefact target, relations and attributes. A simple Hamming distance or Levenshtein distance can be then used to compared two models. We choose to used a more recent metric from Data Mining domain: the cosine dissimilarity [8]. This metric is usualy applied to vectors that represent occurences of word in a text. It is defined as:

$$D_C(\overrightarrow{V_1}, \overrightarrow{V_2}) = 1 - S_C(\overrightarrow{V_1}, \overrightarrow{V_2}) \tag{1}$$

with $S_C(\overrightarrow{V_1}, \overrightarrow{V_2})$ the cosine similarity of two vectors $\overrightarrow{V_1}$ and $\overrightarrow{V_2}$ defined as :

$$S_C(\overrightarrow{V_1}, \overrightarrow{V_2}) = \frac{\overrightarrow{V_1} \cdot \overrightarrow{V_2}}{\|\overrightarrow{V_1}\|\|\overrightarrow{V_2}\|} = \frac{\sum_{i=1}^{n} V_1[i]V_2[i]}{\sqrt{\sum_{i=1}^{n} V_1[i]^2}\sqrt{\sum_{i=1}^{n} V_2[i]^2}} \tag{2}$$

For the simple case where one chromosome contain two models, this distance is quite easy to compute, but in other cases, we obtain a distance matrix. We follow different approaches to try to increase diversity. Indeed, we consider several fitness for the "more than two models" case:

(1) a fitness based on the *lowest* value of the distance matrix: with this approach, we consider that increase the minimal value will normally increase all the distances;
(2) a fitness based on *mean* value of the distance matrix: with this approach, we hope that trying to increase the mean will increase all the distances;
(3) a fitness based on a *sum* of (1) and (2), trying to increase both values;
(4) two different fitness functions: the (1) and the (2), trying to increase one of the two values.

If the three first approaches can be realized with all genetics algorithms, the last approach need the use of a multi-objective evolutionary algorithm. We choose the widely used *NSGA-II* [9, 10].

2.4 Crossover, mutation, feasibility repair and selection

We randomly pick couples of chromosomes and apply a simple one point crossover on them to generate new chromosomes.

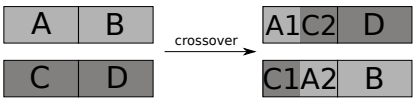

Fig. 1. One point Crossover between two chromosomes composed by 2 models.

We determine that a mutation chance of 0.5‰ for each gene is enough to avoid a too quick convergence and allow diversification. To check feasibility, first we store extra-data, like domains and constraints, to quickly check if new vectors (generated by crossover or mutation) are still valid. Our modified vector is reintroduced in CSP tool to check satisfiability and if necessary, repair it by looking valid instantiation in a classical depth first search exploration.

As selection criterion, when we generating 100 new chromosomes, we keep the 100 best over the overall 200 chromosomes

3 Computational results

We first consider a population with individual composed only by 2 models. In Fig. 2, we can observe that our method provides better results than the CSP approach. Indeed, population at generation 100 has a mean distance score better than the original population. Moreover, we can note that for both distances the lowest value of our final population is better than the higher value of initial one. The lowest values for cosine distance are inherent to the fact that some values of vectors are

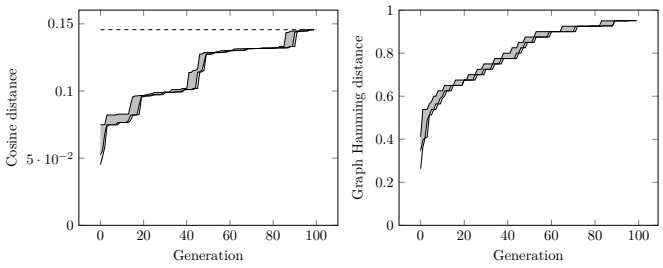

Fig. 2. Evolution of mean distance for population with chromosome with 2 models

too constrained – the dashed line indicates the best possible pairing at last generation, computed with a brute force algorithm.

Fig. 3 shows the different fitness approaches when $p = 5$. Trying min value (1) or mean value (2) brings quite similar results. The combination of two objectives, with sum aggregation (3) and even more with bi-objective vision (4), provides even better results. We also observe that our method provides better results than original one at the cost of an increased computation time to reach convergence.

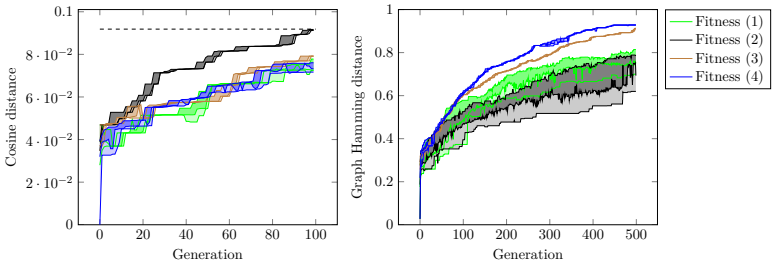

Fig. 3. Evolution of mean distance for population with chromosome composed by 5 models

4 Conclusion

Diversity of artefacts in model generation is an important point. The evolutionary meta-heuristic provide a tool to increase distance between models. With this approach, we are able to increase mean score for more than 50% for considered metrics. As perspectives, we will look at the case where chromosome are mono-model by considering a population fitness, and preliminaries computations seem to provide better results in terms of scalability.

References

1. A. Ferdjoukh, A.-E. Baert, E. Bourreau, A. Chateau, R. Coletta, and C. Nebut. Instantiation of Meta-models Constrained with OCL: a CSP Approach. In *MODELSWARD*, pages 213–222, 2015.
2. J. Cabot, R. Clarisó, and D. Riera. Verification of UML/OCL class diagrams using constraint programming. In *IEEE ICSTW Workshop*, pages 73–80, 2008.
3. C. Alberto G. Pérez, F. Buettner, R. Clarisó, and J. Cabot. EMFtoCSP: A tool for the lightweight verification of EMF models. In *FormSERA*, 2012.
4. J. Cadavid, B. Baudry, and H. Sahraoui. Searching the Boundaries of a Modeling Space to Test Metamodels. In *ICST*, pages 131–140, 2012.
5. J. H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* The University of Michigan Press, 1975.
6. A. Chateau A. Ferdjoukh, E. Bourreau and C. Nebut. A Model-Driven Approach to Generate Relevant and Realistic Datasets. In *SEKE*, 2016.
7. J.-R. Falleri, M. Huchard, M. Lafourcade, and C. Nebut. Metamodel matching for automatic model transformation generation. In *MODELS*, pages 326–340, 2008.
8. A. Singhal. Modern information retrieval: A brief overview. *IEEE Data Engineering Bulletin*, 24(4):35–43, 2001.
9. 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.
10. K. Deb and H. Jain. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014.