Machine Learning for User Modeling in an Interactive Genetic Algorithm for the Next Release Problem

Allysson Allex Araújo and Matheus Paixão

Optimization in Software Engineering Group, State University of Ceará
1700 Paranjana Avenue, 60.714-903, Fortaleza, Brazil
{allyssonallex.dpa,mhepaixao}@gmail.com

Abstract. The Next Release Problem consists in selecting which requirements will be implemented in the next software release. For many SBSE approaches to the NRP, there is still a lack of ability to efficiently include the human opinion and its peculiarities in the search process. Most of these difficulties are due to the problem of the human fatigue. Thus, it is proposed the use of a machine learning technique to model the user and replace him in an Interactive Genetic Algorithm to the NRP. Intermediate results are able to show that an IGA can succesfully incorporate the user preferences in the final solution.

Keywords: Interactive Genetic Algorithm, Machine Learning, Next Release Problem, Search Based Software Engineering.

1 Introduction

During an iterative and incremental software development process, there are some complex problems to deal with, such as selecting which requirements will be implemented in the next release. The term *release* is used to describe a stable and executable version of the system to be delivered to the client, in accordance to his preferences. Given this context, it may be mentioned the widely known Next Release Problem (NRP) [1], which is based on the maximization of the client satisfaction while respecting a predefined budget.

The current mono-objective SBSE approaches to the NRP can be considered as decision-making tools, where the requirements' selection for the next release is automatically made, without an effective participation of the requirements engineer. As a consequence, the users present a certain reluctance in accepting such results. Therefore, it seems that an inclusion of the requirements engineer in the search process can result in a proper strategy to handle the NRP, providing a search technique which is guided by his preferences.

Considering the difficulties in including the decision maker in general optimization techniques, one can highlight the Interactive Optimization. Such strategy is able to employ the human as a protagonist in the solution evaluation process, so that his knowledge and other psychological aspects are incorporated in the search process [2]. This level of human interaction is specially needed when it is difficult to capture the evaluation function through mathematical models or when personal judgements are necessary [3].

The Interactive Evolutionary Computation, which is a branch of the Interactive Optimization, is supported by two key components, which are the human evaluation and computational search through Genetic Algorithms [4]. In spite of the fact that GAs are widespread in SBSE, there is still a lack of ability in using the human preferences in the search process. Thus, the Interactive Genetic Algorithm (IGA) is an alternative to handle this problem. The IGA follows the same concepts of a traditional GA, the difference is regarded to the solution evaluation process, where the solution judgement is performed by the user rather than a mathematical function [5].

Regarding the application of IGAs to requirements engineering problems, it can be pointed out the paper by Tonella et al. [6], which examined the use of an IGA in the requirements prioritization process. Its design aims to minimize the number of requirements pairs evaluations obtained from the user, making the approach more scalable and accurate regarding the requirements classification. It is also interesting to point out the work by Simons et al. [7], where it proposes the use the use of an interactive evolutionary approach alongside with intelligent agents to mitigate the difficulties of software design.

Despite the human involvement being interesting and attractive to the search process, it is also the cause of one of the most critical problems in interactive optimization approaches, which is the human fatigue [5]. This exhaustion occurs due to the repeatedly requests for user evaluations, which ends up being a major threat to the IGA evolution.

Therefore, this paper proposes the use of a machine learning technique to model the requirements engineer preferences during the use of the IGA for the NRP. This way, it will be possible to handle the human fatigue and still incorporate the subjective criteria throughout the search process.

The remaining of this paper is organized as follows: Section 2 specifies the proposed IGA and presents results of the empirical study performed to validate it. Section 3 explains the proposed machine learning modeling for the requirements engineer. Finally, Section 4 concludes and discusses future works.

2 An Interactive Genetic Algorithm for the Next Release Problem

Consider $R = \{r_1, r_2, \ldots, r_N\}$ the set of requirements. Each r_i has an importance value v_i and an effort cost c_i. The NRP model proposed in this work is presented next:

$$\text{maximize: } \alpha.score(X) + \beta.she(X) \tag{1}$$

$$\text{subject to: } cost(X) \leq budget \tag{2}$$

$$\text{where, } score(X) = \sum_{i=1}^{N} v_i x_i \tag{3}$$

$$cost(X) = \sum_{i=1}^{N} c_i x_i \tag{4}$$

where *budget* refers to the release available budget. The decision variable is represented by the vector $X = \{x_1, x_2, \ldots, x_N\}$, so that $x_i = 1$ implies that requirement r_i is included in the next release and $x_i = 0$ otherwise. The *score(X)* function (Equation 3) represents the total importance of the release. Similarly, the *cost(X)* function (Equation 4) represents the total effort cost of the release.

In the IGA application to the NRP, each individual is a release. The requirements engineer provides a "grade" for each individual throughout the IGA evolution. This "grade" is called *subjective human evaluation (she)* and represents the user preferences regarding the requirements selection (Equation 1). In this work, this value is given according to a numerical range previously established. When the release fully satisfies the user, the evaluation is maximum.

The approach used in this paper can be considered as a generalization of the work by Baker et. al [8]. When the weights in Equation 1 are configured to $\alpha = 1$ and $\beta = 0$, the classical NRP is reached.

An empirical study was conducted in order to evaluate the proposed IGA for the NRP. The following topics present the settings and results from this study.

2.1 Empirical Study Settings

The set of instances was randomly generated. The number of requirements varies from 50 to 200. There are no interdependencies between requirements and the importance of each requirement takes an integer value between 1 and 5. The effort cost of each requirement also varies from 1 to 5. The instance name is in the format I_R, where R is the number of requirements.

The *score(X)* value is normalized to the same range of $she(X)$. Such normalization is needed in order to avoid a possible overwhelm regarding the functions *score(X)* and *she(X)*. Thus, the only way to prioritize one of the functions is through the weights α and β.

In order to represent the requirements engineer, a simulator was developed. The main purpose of this simulator is not to faithfully simulate a human being, but rather demonstrate the influence of a certain evaluation profile in the search process. Based on the evaluation profile, the simulator defines a "target-individual", which represents what the requirements engineer would consider as an optimal release. The requirements to be included in the target-individual are chosen based on a certain percentage, which in this particular work was defined as 50%. Three differents evaluation profiles were considered: in the **Random** profile, the requirements are randomly defined. In the **Lower Score**, the requirements with least *score* are included. Similarly, in the **Higher Cost**, the requirements with highest cost are included.

Throughout the IGA, the evaluations for each individual are provided according to the similarity to the target-individual. If the individual's requirements are totally different from the target-individual, the evaluation is minimal. In the other hand, when the individual is equal to the target-individual, the evaluation is maximum. The evaluations are proportionally given for the other possibilities. For this empirical study, the minimum *she(X)* is 0 and the maximum is 10.

Regarding the IGA settings, it was used a fixed amount of 500 individuals, 100 generations, 90% crossover rate, $1/N\%$ mutation rate, an elitism rate of 20% and *budget* equals to 60% of the maximum release cost. All parameters were empirically defined by preliminary tests.

Three weights configurations ($\alpha = 1, \beta = 0$; $\alpha = 0, \beta = 1$; $\alpha = 1, \beta = 1$) were considered. For each weights configuration and instance, the IGA was executed 100 times, collecting the average *similarity degree* of the final solution, which represents how similar is a candidate solution to the target-individual. Consider a set of 6 requirements with a target-individual [100011]. The possible solution [110110], for example, presents 3 equal requirements to the target-individual (r_1, r_3 and r_5), so this solution would present a *similarity degree* of $3/6 = 0.5$. The average of the non-normalized *score(X)* was also collected.

Therefore, the empirical study was conducted in order to answer the following research question:

RQ: What is the influence of the evaluation profile in the search process?

2.2 Results and Analyses

Table 1 shows the average of both *similarity degree* and *score* for each instance, each evaluation profile and different weights settings.

Table 1. Empirical Study Results

Instance	Attributes	RANDOM			LOWER SCORE			HIGHER COST		
		α, β			α, β			α, β		
		1, 0	**0, 1**	**1, 1**	**1, 0**	**0, 1**	**1, 1**	**1, 0**	**0, 1**	**1, 1**
I_50	Similarity Degree	0.48	0.96	0.90	0.30	0.96	0.86	0.30	0.86	0.86
	Score	116.01	71.53	91.55	116.01	60.16	88.96	116.01	63.81	78.21
I_100	Similarity Degree	0.56	0.88	0.81	0.41	0.88	0.79	0.26	0.85	0.80
	Score	230.09	144.63	192.33	230.09	131.84	191.84	230.09	133.56	167.09
I_150	Similarity Degree	0.50	0.79	0.74	0.34	0.79	0.70	0.26	0.76	0.70
	Score	321.55	203.35	307.57	321.60	180.93	287.12	321.50	196.04	260.88
I_200	Similarity Degree	0.46	0.73	0.66	0.29	0.73	0.63	0.26	0.74	0.64
	Score	459.06	299.61	440.65	459.00	259.63	408.31	459.14	266.79	390.04

As can be seen, when the weights are configured to $\alpha = 1$ and $\beta = 0$, all instances present a high *score* value, but a considerably low *similarity degree*. This is due to the fact that only the *score(X)* is considered in the search process.

In contrast, when the weights are configured to $\alpha = 0$ and $\beta = 1$, only *she(X)* is considered and the *similarity degree* is higher for all instances. Thus, the proposed IGA is able to incorporate the requirements engineer preferences in the solutions. Looking at the I_50 and I_100 instances, with the Random profile, the solutions present a *similarity degree* of 0.96 and 0.88, respectively. In this configuration, it is also clear the *score* values tend to be lower than the previous ones. This is due to the fact that in most cases the solution the requirements engineer considers as good, does not necessarily present a high *score*.

However, the presented approach also allows the configuration $\alpha = 1$ and $\beta = 1$ which aims at optimizing the user preferences and the *score* simultaneously. Such weights configuration can provide valuable insights regarding the trade-offs between *similarity degree* and *score*. This behavior can be seen in the instances I_150 and I_200 for the Lower Score profile, which present a *similarity degree* of 0.70 and 0.63, and *score* of 287.12 and 487.31, respectively.

The Lower Score and Higher Cost profiles are unusual in a real software development environment, but the proposed IGA could still incorporate these preferences in the final solution. Given these results, it is stated the final solutions are considerably influenced by the evaluations profiles, in a way they tend to get closer to the target-individuals, answering the research question.

3 A Machine Learning Approach for User Modeling

As demonstrated in the previous section, the IGA is capable of incorporating the user preferences in the search process. However, as explained earlier, the human fatigue problem makes an interactive approach unfeasible when the number of interactions is high. For an IGA settings with 500 individuals and 100 generations, for example, the requirements engineer would be asked 50000 times.

In order to handle this difficulty, this paper proposes a machine learning technique to model the user evaluation profile. Thus, the learning model would use the individuals, and the respective human evaluations, as a training set, replacing the requirements engineer after a while. The architecture of the proposed machine learning model alongside the IGA can be seen in Figure 1.

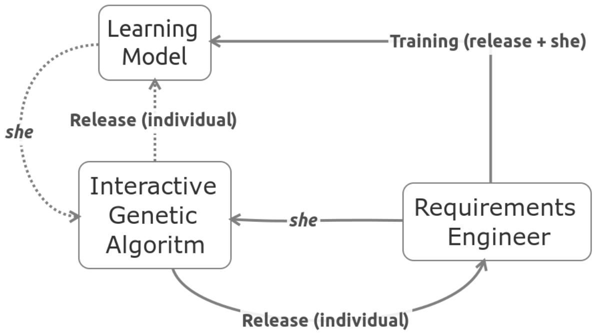

Fig. 1. The Architecture of the Learning Model alongside the IGA

The process is divided into two distinct stages. In the first stage (solid lines), all individuals will be evaluated by the requirements engineer. The IGA is guided by the user's preferences while the learning model learn its behavior. The learning model will be trained until a confidence level is satisfied or a certain number of evaluations is reached. In the second stage (dotted lines), the model would have already learnt the user preferences, and the remaining evaluations will be fully transferred to it, resulting in a significant reduction in human fatigue.

Therefore, when incorporating a learning model to the IGA, it is expected the results remain consistent to the evaluation profile, but with a significant decrease in the number of questions to the requirements engineer.

Currently, the learning model is under development and it is expected to be tested with different learning techniques (Least Means Square, Multilayer Perceptron, etc) which one present a better suiting to the proposal.

4 Conclusion

The requirements selection is a complex task in an iterative and incremental software development project. When search techniques are used to tackle such problems, it becomes difficult to incorporate the user's preferences to the process.

The objective of this work was to develop a feasible Interactive Genetic Algorithm to the Next Release Problem, but in a way it could also deal with the human fatigue. In order to handle this problem, it is proposed a machine learning technique to model the requirements engineer preferences and replace him throughout the search process. Intermediate IGA results show that it is able to incorporate the requirements engineer knowledge in the final solutions.

As future works, it is expected to finalize the implementation and tests related to the learning model; assess different ways in which the requirements engineer can evaluate the solutions; consider interdependencies between requirements; apply the IGA for the multiobjective version of the NRP.

References

1. Bagnall, A.J., Rayward-Smith, V.J., Whittley, I.M.: The next release problem. Information and Software Technology 43(14), 883–890 (2001)
2. Harman, M., Mansouri, S.A., Zhang, Y.: Search based software engineering: A comprehensive analysis and review of trends techniques and applications. Department of CS, Kings College London, Tech. Rep. TR-09-03 (2009)
3. Harman, M., McMinn, P., de Souza, J.T., Yoo, S.: Search based software engineering: Techniques, taxonomy, tutorial. In: Meyer, B., Nordio, M. (eds.) LASER Summer School 2008-2010. LNCS, vol. 7007, pp. 1–59. Springer, Heidelberg (2012)
4. Harman, M.: Search based software engineering for program comprehension. In: 15th IEEE International Conference on Program Comprehension, ICPC 2007, pp. 3–13. IEEE (2007)
5. Takagi, H.: Interactive evolutionary computation: Fusion of the capabilities of ec optimization and human evaluation. Proceedings of the IEEE 89(9), 1275–1296 (2001)
6. Tonella, P., Susi, A., Palma, F.: Using interactive ga for requirements prioritization. In: 2010 Second International Symposium on Search Based Software Engineering (SSBSE), pp. 57–66. IEEE (2010)
7. Simons, C.L., Parmee, I.C., Gwynllyw, R.: Interactive, evolutionary search in upstream object-oriented class design. IEEE Transactions on Software Engineering 36(6), 798–816 (2010)
8. Baker, P., Harman, M., Steinhofel, K., Skaliotis, A.: Search based approaches to component selection and prioritization for the next release problem. In: 22nd IEEE International Conference on Software Maintenance, ICSM 2006, pp. 176–185. IEEE (2006)