Learning from evolved next release problem instances

Zhilei Ren, He Jiang, Jifeng Xuan, Shuwei Zhang, Zhongxuan Luo

Learning from Evolved Next Release Problem Instances

Zhilei Ren He Jiang
Dalian University of Technology
Dalian 116623, China
{zren, jianghe}@dlut.edu.cn

Jifeng Xuan
INRIA Lille – Nord Europe
Lille 59650, France
jifeng.xuan@inria.fr

Shuwei Zhang Zhongxuan Luo
Dalian University of Technology
Dalian 116623, China
zhangshuwei@mail.dlut.edu.cn
zxluo@dlut.edu.cn

ABSTRACT

Taking the Next Release Problem (NRP) as a case study, we intend to analyze the relationship between heuristics and the software engineering problem instances. We adopt an evolutionary algorithm to evolve NRP instances that are either hard or easy for the target heuristic (GRASP in this study), to investigate where a heuristic works well and where it does not, when facing a software engineering problem. Thereafter, we use a feature-based approach to predict the hardness of the evolved instances, with respect to the target heuristic. Experimental results reveal that, the proposed algorithm is able to evolve NRP instances with different hardness. Furthermore, the problem-specific features enables the prediction of the target heuristic's performance.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/ Specifications—*Methodologies*

Keywords

Genetic algorithm, Next release problem, Problem hardness

1. INTRODUCTION

In this study, we take the Next Release Problem (NRP) as a case study, to analyze the relationship between heuristic algorithms and the software engineering problem instance hardness. We are interested in investigating both the strength and the weakness of a given heuristic, when facing a software engineering problem. To achieve this, we use a genetic algorithm to evolve a set of NRP instances that are hard and easy to solve, respectively, with respect to the target heuristic. Then, we extract a set of features from the evolved instances, to capture the distribution information of the instance specific variables. With these proposed features, we are able to predict the target heuristic's performance.

2. THE NEXT RELEASE PROBLEM

In a software project, let R be the set of candidate requirements with $|R| = m$. Each requirement $r_j \in R(1 \leq j \leq m)$ corresponds to a nonnegative cost $c_j \in C$. A directed acyclic graph $G = (R, E)$ indicates the dependencies between these requirements, with R denoting the vertices and E specifying the arcs. In the dependency graph G, an arc $(r', r) \in E$ means that the requirement r relies on r'. We call the requirement r the child requirement of r'. $parents(r)$ is defined as $parents(r) = \{r' \in R | (r', r'') \in E, r'' \in parents(r)\}$. All requirements of $parents(r)$ have to be realized to ensure that r is implemented.

Let S be the customer set with $|S| = n$. Each customer $s_i \in S$, requests a subset of requirements $R_i \subseteq R$. Let $w_i \in W$ be the weight for the customer s_i. Let $parents(R_i)$ be $\bigcup_{r \in R_i} parents(r)$. Given a customer s_i, let the set of requirements requested by s_i be $\hat{R}_i = R_i \bigcup parents(R_i)$. Hence, a customer s_i can be satisfied if and only if all the requirements in R_i are realized during the next release. Let the cost of satisfying the customer s_i be $cost(\hat{R}_i) = \sum_{r_j \in \hat{R}_i} c_j$. The NRP is formulated as follows. Given a set of customers $\{y_1, \cdots, y_n\}$ and a set of requirements $\{x_1, \cdots, x_m\}$, as well as the corresponding weights $\{w_1, \cdots, w_n\}$ and cost $\{c_1, \cdots, c_m\}$, the NRP aims to select a subset of customers, to maximize the corresponding weights. Without loss of generality, in this study we assume that the customers are sorted in descending order with respect to their weight-to-cost ratios ($\frac{w_1}{cost(\hat{R}_1)} > \frac{w_2}{cost(\hat{R}_2)} > \cdots > \frac{w_m}{cost(\hat{R}_m)}$). The objective of the NRP is to maximize the following formulations:

$$\sum_{i=1}^{m} w_i y_i \tag{1}$$

subject to

$$\sum_{i=1}^{n} c_i x_i \leq B, \tag{2}$$

$$x_i \geq x_j, \forall (r_i, r_j) \in E, \tag{3}$$

$$x_i \geq y_j, \forall i, j \text{ where } r_i \text{ is required by customer } j. \tag{4}$$

$$x_i, y_j \in \{0, 1\}, 1 \leq i \leq n, 1 \leq j \leq m. \tag{5}$$

3. EVOLUTIONARY ALGORITHM TO GENERATE NRP INSTANCES

A genetic algorithm is employed to evolve NRP instances, which are hard or easy to solve, respectively. The algorithm maintains a population of 20 individuals. Each individual is represented by an NRP instance. More specifically, the variables of each individual involve the values of the weights and the costs. The other instance specific parameters, such as the number of customers, the budget bound, and the dependency relationships are inherited from the input seed instance. For the genetic operators, we employ the single point crossover, and the uniform mutation with the mutation rate 0.05. For the selection method, we adopt the truncation selection. The fitness value of a given instance is assigned

by the mean error (compared against the optimal solution), which is achieved by 30 independent runs of GRASP [2].

4. CAPTURING THE NRP CHARACTERISTICS USING FEATURES

We consider two classes of features that may influence the instance hardness. The first set of features include the correlation between each customer's weight and the corresponding costs, the standard deviation of the customers' weights, the standard deviation of the costs, the standard deviation of the weight-to-cost ratio, the coefficient of variation of all the customers' weights, the coefficient of variation of the costs, and the coefficient of variation of the weight-to-cost ratio. The second set of features are extracted from the relaxed version of the NRP. The motivation of these features are inspired by the core concept from the knapsack problem literatures [4]. As mentioned, in this study, we assume the customers are sorted according to the weight-to-cost ratio. Furthermore, due to the intrinsic difficulty of the NRP, we consider the relaxed version of the NRP in [2], i.e., we replace constraint (5) with the following constraint:

$$x_i \in [0,1], y_j \in [0,1], 1 \leq i \leq n, 1 \leq j \leq m. \quad (6)$$

Given the optimal solution $\{\hat{y}_1, \cdots, \hat{y}_m\}$ to the relaxed NRP, we define $\hat{j}_1$, $\hat{j}_2$, $\hat{j}_{min}$, $\hat{j}_{max}$ as follows: $\hat{j}_1 = \min\{1 \leq j \leq m | \hat{y}_j = 0\}$, $\hat{j}_2 = \max\{1 \leq j \leq m | \hat{y}_j > 0\}$, $\hat{j}_{min} = \min\{\hat{j}_1, \hat{j}_2\}$, $\hat{j}_{max} = \max\{\hat{j}_1, \hat{j}_2\}$. With the auxiliary variables $\hat{j}_{min}$ and $\hat{j}_{max}$, the customer set could be partitioned into three subsets (corresponding to customers with index ranging from $[1, \hat{j}_{min})$, $[\hat{j}_{min}, \hat{j}_{max}]$, and $(\hat{j}_{max}, m]$). The rescaled sum of weights (σ_1–σ_3) and the rescaled sum of costs (δ_1–δ_3) are defined as follows: $\sigma_1 = \frac{\sum_{1 \leq i < \hat{j}_{min}} w_i}{\sum_{1 \leq i \leq m} w_i}$, $\sigma_2 = \frac{\sum\limits_{\hat{j}_{min} \leq i \leq \hat{j}_{max}} w_i}{\sum\limits_{1 \leq i \leq m} w_i}$, $\sigma_3 = \frac{\sum\limits_{\hat{j}_{max} < i \leq m} w_i}{\sum\limits_{1 \leq i \leq m} w_i}$, $\delta_1 = \frac{\sum\limits_{1 \leq i < \hat{j}_{min}} cost(\hat{R}_i)}{\sum\limits_{1 \leq i \leq n} c_i}$, $\delta_2 = \frac{\sum\limits_{\hat{j}_{min} \leq i \leq \hat{j}_{max}} cost(\hat{R}_i)}{\sum\limits_{1 \leq i \leq n} c_i}$, $\delta_3 = \frac{\sum\limits_{\hat{j}_{max} < i \leq m} cost(\hat{R}_i)}{\sum\limits_{1 \leq i \leq n} c_i}$. Also, the upper bound obtained from the optimal solution to the relaxed NRP instance is used as a feature.

5. EXPERIMENTAL RESULTS

To predict the target heuristic's performance, we adopt the R port of random forest model **randomForest** [3]. We employ the open source package SCIP [1] as the solver to obtain the optimal solutions. For the seed instances, we use the benchmark instances from [2]. To evolve hard NRP instances, we execute the algorithm over each seed instance for 10 independent runs. The easy NRP instances are generated similarly. Hence, we obtain 150 hard NRP instances and 150 easy NRP instances, respectively. Furthermore, for both the hard and the easy instances, we randomly select 50% of the instances as the training instances, and treat the rest of the instances as the test instances.

We first investigate the distribution of the mean error of the target heuristic over the two sets of evolved instances. We notice that the median difficulty of the hard instance set is significantly higher than the easy instance set. Besides, we find that the mean error achieved by the hard instance set has a larger variation (ranging from around 0.1 to above 0.5), while the mean error for the easy instance set has a relative small variation. Then, we investigate the possibility of predicting the solution quality. To achieve this, we use the random forest regression model. Fig. 1 presents the results of the regression model. In the figure, the x-axis and the y-axis indicate the predicted mean error and the mean error achieved by the target heuristic. Each point indicates the results over one or more instances. From the figure, we observe that the regression model is able to achieve good prediction quality. Most points in the figure lie around the reference line ($y = x$). Furthermore, the Root Mean Squared Error obtained by the regression model is 0.0409.

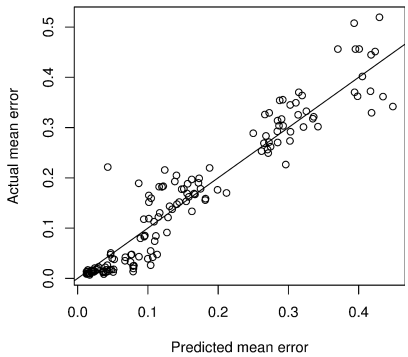

Figure 1: Comparison between the predicted performance and the actual performance

6. CONCLUSIONS AND FUTURE WORK

In this paper, we focus on evolving hard and easy requirements engineering problem instances. Given the target heuristic (GRASP), we develop an evolutionary algorithm to generate hard and easy NRP instances, respectively. We also investigate the possibility of predicting the performance of the target heuristic over the evolved NRP instances. For the future work, we are interested in leveraging the problem-specific features to guide the algorithm design process, using approaches such as hyper-heuristics [5].

7. REFERENCES

[1] T. Achterberg. SCIP: solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[2] A. J. Bagnall, V. J. Rayward-Smith, and I. M. Whittley. The next release problem. *Information and Software Technology*, 43(14):883–890, 2001.

[3] A. Liaw and M. Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[4] D. Pisinger. Core problems in knapsack algorithms. *Operations Research*, 47(4):570–575, 1999.

[5] Z. Ren, H. Jiang, J. Xuan, and Z. Luo. Hyper-heuristics with low level parameter adaptation. *Evolutionary computation*, 20(2):189–227, 2012.