

# A stability assessment of solution adaptation techniques for analogy-based software effort estimation

Passakorn Phannachitta<sup>1</sup> · Jacky Keung<sup>2</sup> ·  
Akito Monden<sup>3</sup> · Kenichi Matsumoto<sup>1</sup>

Published online: 31 May 2016  
© Springer Science+Business Media New York 2016

**Abstract** Among numerous possible choices of effort estimation methods, analogy-based software effort estimation based on Case-based reasoning is one of the most adopted methods in both the industry and research communities. Solution adaptation is the final step of analogy-based estimation, employed to aggregate and adapt to solutions derived during the case-based reasoning process. Variants of solution adaptation techniques have been proposed in previous studies; however, the ranking of these techniques is not conclusive and shows conflicting results, since different studies rank these techniques in different ways. This paper aims to find a stable ranking of solution adaptation techniques for analogy-based estimation. Compared with the existing studies, we evaluate 8 commonly adopted solution techniques with more datasets (12), more feature selection techniques included (4), and more stable error measures (5) to a robust statistical test method based on the Brunner test. This comprehensive experimental procedure allows us to discover a stable ranking of the techniques applied, and to observe similar behaviors from techniques with similar adaptation mechanisms. In general, the linear adaptation techniques based on the functions of size and productivity (e.g., regression towards the mean technique) outperform the other techniques in a more robust experimental setting adopted in this study. Our empir-

---

Communicated by: Martin Shepperd

---

✉ Passakorn Phannachitta  
phannachitta-p@is.naist.jp

✉ Jacky Keung  
Jacky.Keung@cityu.edu.hk

<sup>1</sup> Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan

<sup>2</sup> Department of Computer Science, City University of Hong Kong, Hong Kong, China

<sup>3</sup> Graduate School of Natural Science and Technology, Okayama University, Okayama, Japan

ical results show that project features with strong correlation to effort, such as software size or productivity, should be utilized in the solution adaptation step to achieve desirable performance. Designing a solution adaptation strategy in analogy-based software effort estimation requires careful consideration of those influential features to ensure its prediction is of relevant and accurate.

**Keywords** Software effort estimation · Analogy-based estimation · Solution adaptation techniques · Ranking instability · Robust statistical method

## 1 Introduction

Analogy-based estimation is a widely used approach to estimate the amount of software development effort required in a software project. This approach is used mainly because of its excellent estimation performance and its intuitive appeal to practitioners (Keung 2009). The principle of estimation by analogy is based on the hypothesis that software projects with similar characteristics should require a similar amount of effort to complete (Shepperd and Schofield 1997). Thus, the estimation procedure for a new project consists of a retrieval of the similar historical projects and then reuse of the retrieved efforts. One of the major issues of conventional analogy-based estimation is that, even though the most similar past projects are correctly retrieved, their characteristics often deviate somewhat from the new project (Jørgensen et al. 2003; Li et al. 2007, 2009). Therefore, correctly adjusting the reused effort is necessary to provide an accurate estimate.

The imperative procedure to adjust the effort is known as solution adaptation (case adaptation in Case-based reasoning). It procures a solution estimate from a set of similar and known past project cases by evaluating the degree of difference between similar retrieved projects and the new project to be estimated, and refining the estimated effort based on that degree difference (Li et al. 2009). To date, various solution adaption techniques have been proposed such as linear size adaptation, non-linear adaptation based on neural networks, and many others (Chiu and Huang 2007; Jørgensen et al. 2003; Kirsopp et al. 2003; Li et al. 2007, 2009; Walkerden and Jeffery 1999); however, the performance of many techniques were reported differently in different studies, producing conflicting findings. For example, in a more recent replicated study by Azzeh (2012) covering 8 most commonly adopted techniques in practice, the replicated evaluation of a technique based on neural networks produced results that contradict those of another study by Li et al. (2009). Azzeh also suggested that there is no single best technique; most of the techniques are still a long way from reaching optimal solutions, and some techniques perform well in a certain dataset and at a certain configuration (e.g. a fixed number of similar projects to retrieve), while they do not in any other cases.

Based on these conflicting results and inconsistent behaviors observed in many solution adaptation techniques in recent studies, we recognize that the conclusion instability problem exists and speculate that weaknesses of the experimental method were the main causes of the issue. As suggested by Menzies et al. (2010), the precise stable results require well-controlled experimental conditions of (1) the method to generate training/test instances, (2) the performance evaluation criteria used, and (3) the datasets used. We then observed these weaknesses in the existing studies and found many studies of solution adaption techniques had applied different feature subset selection methods, with different performance measures, and had recorded the performance regarding different fixed numbers of similar projects to retrieve. Additionally, we also observed unconvincing experimental designs such

as the use of malformed datasets, and the misuse of deprecated evaluation criteria such as *MMRE*, despite proof that *MMRE* has been a biased and untrustworthy measure for over a decade (Foss et al. 2003; Kitchenham and Mendes 2009). We believe that a more comprehensive experimental design than that of the existing studies may be required to discover a more stable conclusion of different performances of solution adaptation techniques, an important study that should provide a reliable guideline for identifying superior techniques to build estimation models.

Our aim in this study, therefore, is to determine a stable ranking of different effort adaptation techniques commonly adopted in analogy-based software effort estimation, through a comprehensive experimental procedure that has been used to draw conclusion stability in the software effort estimation literature. As suggested by Menzies et al. (2010) and Keung et al. (2013), conclusion stability in software effort estimation requires sufficiently large variants of datasets, dataset preprocessors, evaluating criteria, and also, other control factors according to the focus and scope of particular studies. Our hypotheses in this study are that (1) with thorough comparisons of the widely used solution adaptation techniques for analogy-based estimation, undertaken by a comprehensive evaluation method, and evaluated with a robust statistical test method, we can assess a stable ranking list of solution adaptation techniques commonly adopted for analogy-based estimation. In addition, (2) if we are able to assess and rank all the adaptation techniques correctly, we can observe similar behavior of similar adaptation techniques in other settings, such as techniques with similar adaptation mechanisms that should have similar accurate performances.

Specifically, this study highlights the novelty in the adopted comprehensive experimental methodology and the conclusion stability of solution adaptation techniques produced from the experiments and analyses based on this methodology. In a short summary, the present study makes several noteworthy points as follows:

- The main evaluation framework used in this study is our improvements to the Stable ranking indication method proposed by Keung et al. (2013) in two main components:
  - Our evaluation criteria replaces the list of measures being criticized as biased measures with 5 stable error measures (MAR, MdAR, SD, LSD, and RSD), in which their stability has been statistically validated in Foss et al.’s study on evaluation criteria (Foss et al. 2003).
  - The performance recorded by the 5 error measures is subject to Brunner et al.’s test (Brunner et al. 2002), one of the most robust non-parametric test method. This test method was strongly suggested to be more robust and reliable than Wilcoxon rank-sum test (Demšar 2006) used in the proposed study of Keung et al. (2013).
- In place of using only a single feature subset selection method versus using the entire feature set, as commonly seen in the literature (Azzeh 2012), our experiment adopts a total of 4 methods in order to avoid possible bias, if the only selected method tends to favor some solution adaptation techniques over others.
- For each particular training/test instance, we tune the number of similar retrieved projects (the parameter  $k$ ) and use the value of  $k$  that provides the most fit to the training instance (Baker’s *best-k* (Baker 2007)).
- To compensate for a lack of number of datasets to draw stable conclusions, we use 12 datasets from the tera-PROMISE repository (Menzies et al. 2015), where 3 of which are homogenized datasets being decomposed from the Nasa93 dataset.

- Leveraged by this comprehensive experimental methodology, we are able to assess a stable ranking of the 8 adaptation techniques, commonly adopted in practice (Azzeh 2012).

Our further analyses of results show that we found some results that agreed with existing studies but others that contradicted previous results. For example, we agree with Azzeh (2012) that neural networks consistently perform poorly, but, in contrast, we found that genetic algorithms also perform poorly. Hence, the contributions of this study are not limited only to a comprehensive ranking list of solution adaptation techniques itself, but hopefully the study also encourages researchers or practitioners to seek a robust and consistent methodology when undertaking a research study of this kind.

Our primary motivations and experiments are organized to answer these following research questions:

RQ1 How stable is the ranking list we discovered?

RQ2 Which are the high-performance solution adaption techniques for analogy-based estimation?

RQ3 What is the overall performance of each solution adaption technique?

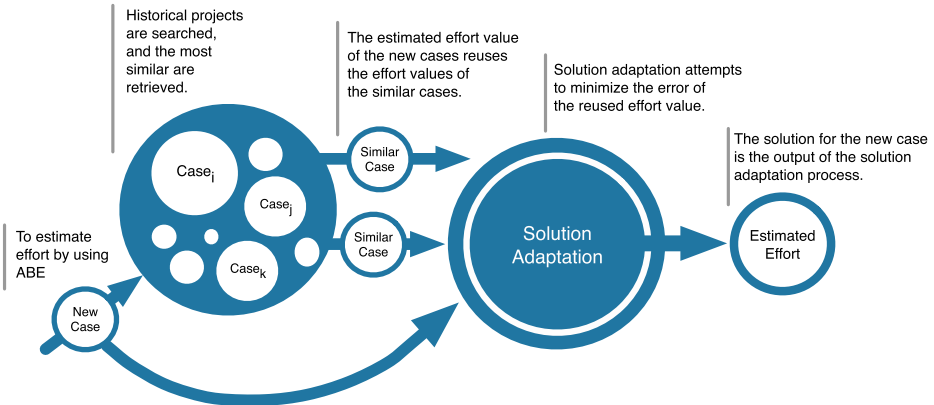
This paper is structured as follows: Section 2 presents the background and related work about analogy-based estimation and the importance of solution adaptation techniques, and briefly overviews an assessment of the stable ranking of estimation methods. Section 3 presents the methodologies that generate a stable ranking list. Section 4 presents the results regarding each research question. Section 5 and Section 6 discuss the findings and threats to the validity of this study, respectively. Finally, Section 7 concludes this paper.

## 2 Background

### 2.1 Analogy-Based Effort Estimation (ABE)

Analogy-based estimation is a reasoning-based estimation method, which estimates an amount of effort for a new project case following a hypothesis such that software projects with similar characteristics should require a similar amount of effort to complete. When a new case arrives for an estimate, the case is treated as a query for retrieving a set of the past project cases that are the most similar to the new case. Without any adaptation applied, the new case will directly reuse the effort value of the most similar past project as its estimated effort; otherwise, solution adaptation is applied to the retrieved multiple similar solutions from the selected projects, and to refine and finally arrive an estimated effort solution that is most appropriate to the new case. Figure 1 graphically explains an estimation by analogy when solution adaptation is being applied, focusing from the arrival of a new case up to produce the estimated effort value for the case.

The name *ABEO-kNN* was coined by Keung et al. (2013) as a basic and standard form of an ABE framework. *ABEO-kNN* uses a variant of Euclidean distance function that combines with the Hamming distance function (Wilson and Martinez 1997) to calculate the similarity between project cases, and uses  $k$  nearest neighbor (kNN) technique to select the  $k$  software project cases that are most similar to the new case. The default solution adaptation technique of *ABEO-kNN* is an unweighted mean of the effort values of the  $k$  software projects selected by kNN. The  $k$  parameter value is commonly fixed as a static value in the literature. For example, Walkerden and Jeffery (1999) set the number statically to 1 in their experiment.



**Fig. 1** A graphical explanation of the analogy based estimation process

Kirsopp and Shepperd observed  $k = 2$  in Kirsopp et al. (2003). Mendes et al. examined the number in a range from 1 to 3 in Mendes et al. (2003) to be appropriate. Li et al. used  $k = 5$  in the study discussed in Li et al. (2009). However, Baker (2007) tuned the  $k$  parameter value by a method called wrapping and used the *best-k* value that fits the best to the training set.

In our study, we adopt only the method developed by Baker (2007), which is considered the most optimal procedure for such an operation. We denote the name *ABEO-best-kNN* as the basic *ABEO-kNN* having the  $k$  parameter selected by the Baker's *best-k* approach, and adopt the *ABEO-best-kNN* as the baseline framework used in this paper to evaluate the performance solution adaptation technique.

## 2.2 Solution (Case) Adaptation Techniques

Compared with an analogy-based estimation process without any solution adaptation applied, an effort value estimated by a process in which the estimated effort value was adjusted by any adaptation technique is on average more robust and reliable as stated by a number of studies, such as in (Azzeh 2012; Li et al. 2007, 2009; Walkerden and Jeffery 1999). This is because the effort used in similar retrieved projects often deviate from the new project case. In this study we replicate 8 solution adaptation techniques, often appearing in analogy-based estimation studies (Azzeh 2012; Chiu and Huang 2007; Jørgensen et al. 2003; Kirsopp et al. 2003; Li et al. 2007, 2009; Walkerden and Jeffery 1999). The list of 8 techniques are the choices of our study because they were recently replicated in the study by Azzeh (Azzeh 2012), in which several inconsistent behaviors among the adaptation techniques were observed. In the study of Azzeh (2012), these 8 techniques were evaluated based on 7 datasets, and the author concluded that most adaptation techniques are still long way from reaching the optimal solutions, because their behaviors are inconsistent with those techniques that performed well in certain selected studies, but did not consistently performed well in other studies. Table 1 lists and summarizes the 8 adaptation techniques used in this study, and a detailed explanation of the 8 techniques is as follows:

**Adaptation Technique #1** Unweighted average of the effort (*UAVG*) is a primitive adaptation technique to adjust the effort value for the basic *ABEO-kNN* (Shepperd and Schofield

**Table 1** Summary of 8 adaptation techniques included in this study

Abbrev.	Solution adaptation techniques	Adjustment function	Adjustment feature
1 UAVG	Unweighted mean of the $k$ analogues (Shepperd and Kadoda 2001)	Arithmetic mean	Effort
2 IRWM	Inverse-rank weighted mean of the $k$ analogues (Mendes et al. 2003)	Arithmetic mean	Effort
3 LSA	Linear size adaptation (Walkerden and Jeffery 1999)	Linear	Size feature
4 MSA	Multiple size adaptation (Kirsopp et al. 2003)	Linear	Size feature(s)
5 RTM	Regression towards the mean (Jørgensen et al. 2003)	Linear	Size feature
6 AQUA	Similarity-based adaptation (Li et al. 2007)	Linear	All features
7 GA	Adaptation based on Genetic algorithm (Chiu and Huang 2007)	Linear	All features
8 NNet	Non-linear adaptation based on Neural networks (Li et al. 2009)	Non-linear	All features

1997; Keung et al. 2013). *UAVG* aggregates effort values of the selected  $k$  analogues (neighbours) by using an unweighted mean:

$$Effort(P_{new}) = \frac{1}{k} \sum_{i=1}^k Effort(P_{analogue_i}), \quad (1)$$

where  $k$  is the total number of project analogues, and  $Effort(P_{new})$  is the estimated effort for the new case.

**Adaptation Technique #2** Inverse-rank weighted mean (*IRWM*) applies different weight values to different project analogues. The weight used in *IRWM* is based on the similarity ranking between each project analogue and the new case, as the technique assumes that project analogues that are closer to the new case are more important (Mendes et al. 2003). The *IRWM* formula is depicted in (2).

$$Effort(P_{new}) = \frac{1}{\sum_{i=1}^k i} \sum_{i=1}^k ((k-i+1) \times Effort(P_{analogue_i})) \quad (2)$$

where  $i = 1$  refers to the nearest analogue, and  $i = k$  refers to the most distant analogue to the new case.

**Adaptation Technique #3** Linear size adaption (*LSA*) is based on the linear extrapolation between a software size variable of the new case and of its retrieved similar projects (Walkerden and Jeffery 1999). The software size variable is, for example, Adjusted Function Points, Raw Function Points, and Lines of code. Walkerden and Jeffery (1999), based on their observations, suggested that adjusting the effort using a size variable can provide a robust estimate because software size always has a strong correlation with the effort variable. Furthermore, adjusting the effort value using software size allows the estimation to scale the retrieved effort value up or down to the expected size of the new case, if the new

case is planned to be developed in a different size from that of its similar projects. Equation 3 depicts the formula of the *LSA* technique.

$$Effort(P_{new}) = \frac{Size(P_{new})}{\frac{1}{k} \sum_{i=1}^k Size(P_{analogue_i})} \times \frac{1}{k} \sum_{i=1}^k Effort(P_{analogue_i}) \quad (3)$$

Note that in the case where a dataset consists of multiple variables indicating software size, the single software size variable selected in the experiment is the one that exhibits the highest correlation coefficient with the effort.

**Adaptation Technique #4** Multiple size adaptation (*MSA*) extends the *LSA* technique to handle the case in which software size is described by multiple arbitrary attributes. These cases are commonly seen in web application development (Kirsopp et al. 2003). *MSA* aggregates multiple size variables using a mathematical mean, and then applies the mean of the size variables to the *Size* terms in (3).

**Adaptation Technique #5** Regression toward the mean (*RTM*) calibrates the productivity of project cases in which the productivity value may not consistent with each of the other project cases (Jørgensen et al. 2003) before applying the calibrated productivity to adjust the effort. The essential hypothesis for the productivity calibration follows the statistical phenomenon, known as *Regression toward the mean*, stating that, any extreme instance on its first measurement will tend to move toward the population mean on its second measurement. In the estimation process, projects are divided into groups by a single categorical variable, selected prior to the estimation. Then, the productivity of the similar retrieved projects are adjusted towards the average productivity among projects in the same group. The adjusted productivity is then calculated into the estimated effort by following the triangular relation of the  $Effort = Size \times Productivity$  (Walkerden and Jeffery 1999). The entire formula for the *RTM* technique is depicted in (4).

$$Effort(P_{new}) = Size(P_{new}) \times \left( \frac{1}{k} \sum_{i=1}^k Productivity(P_{analogue_i}) + \left( M - \frac{1}{k} \sum_{i=1}^k Productivity(P_{analogue_i}) \right) \times (1 - r) \right) \quad (4)$$

where  $M$  is the mean productivity of the project in the same group, and  $r$  is the correlation coefficient between the productivity of the project analogues and the actual productivity of the projects in the same group.

In the experiments, there are two important parameters for *RTM* to be configured: a single software size variable and a categorical variable used to divide datasets into multiple coherence groups. For the single size variable, we used the same criteria as for *LSA*. That is the selected size variable is the one that exhibits the highest correlation coefficient with the effort. For the other categorical variable, unfortunately, our literature review found none of the existing studies discussed the selection criteria when there are more than one categorical variables being presented in a dataset. Hence, we adopted simple criteria as follows:

- In the case where no categorical variable existed in a dataset, we treated the entire dataset as a single coherent data group.
- In the case where there is one or more categorical variables existed in a dataset, the categorical variable of our choice was the one that generated coherence groups with

the minimum sum of pairwise difference of the effort, calculated across all the divided coherence groups.

**Adaptation Technique #6** Similarity-based adaptation (*AQUA*) adjusts the estimated effort value by the aggregated degree of similarity between pairs of project features of the new case and its retrieved similar projects. The degree of similarity is an inverse of distance, which is used in identifying similar projects of the new case. The aggregation formula involves the sum of the product of the normalized *global similarity* degree as depicted in (5). Note that the term *global similarity* refers to the similarity between the paring of the new case and its analogues, measured based on all project features. In addition, *Global similarity* is the product of the sum of the *local similarity*, which is referred to as the similarity between paring of the new case and its similar projects, measured based on only a single project feature.

$$Gsim(P_{new}, P_{analogue_i}) = \sum_{j=1}^f Similarity(Feature_j(P_{new}), Feature_j(P_{analogue_i}))$$

$$Effort(P_{new}) = \frac{1}{k} \times \frac{\sum_{i=1}^k (Gsim(P_{new}, P_{analogue_i}) \times Effort(P_{analogue_i}))}{\sum_{i=1}^k Gsim(P_{new}, P_{analogue_i})}, \quad (5)$$

where *Gsim* is the *global similarity*,  $Feature_j(P_{new})$  is the value of feature  $j$  of the new case, and  $k$  is the number of project analogues of the new case.

**Adaptation Technique #7** Adaptation based on Genetic algorithm (*GA*), proposed by Chiu and Huang (2007), adopts a Genetic algorithm method to adjust the effort values retrieved from project analogues. The adaptation mechanism is similar to *AQUA*, which is based on the similarity degrees between the new case and its similar past projects. One genetic algorithm model is adopted for one single project feature to derive a suitable linear model that approximates the effort difference by *local similarity* (i.e. the same notion as described in *AQUA*). Then *GA* aggregates each optimized linear models by the sum of its products into a single linear equation for adjustment, and applies the aggregated model to effort in additive form as shown in (6).

$$e(P_{new}, P_{analogue_i}) = \sum_{j=1}^f \beta_j \times Similarity(Feature_j(P_{new}), Feature_j(P_{analogue_i}))$$

$$Effort(P_{new}) = \frac{1}{k} \sum_{i=1}^k (Effort(P_{analogue_i}) + e(P_{new}, P_{analogue_i})), \quad (6)$$

where  $e(P_{new}, P_{analogue_i})$  is an aggregation of the optimized linear models, each of which is optimized by a Genetic algorithm based on one single project feature, and  $\beta_j$  is the coefficient of  $Feature_j$  and  $Effort$  which is learned by the Genetic algorithm. According to the replicated study by Azzeh (2012), this adaptation techniques is on average the most accurate.

In our experiment, we optimized the genetic algorithm based on an absolute error measure called MAR, i.e., the absolute difference between actual and predicted effort values. This optimization criteria are different from both the proposed study by Chiu and Huang



(2007) and the replicated study by Azzeh (2012), which relied on error measures named MMRE and Pred. These two error measures were strongly criticized as being the measures no longer believed to be reliable and trustworthy (Foss et al. 2003; Kitchenham and Mendes 2009).

**Adaptation Technique #8** Non-linear Adaptation based on neural networks (*NNet*) proposed by Li et al. (2009), trains a neural network to capture the degree of difference of project features between pairs of project cases. Then the degree of difference is converted to the amount of effort difference in the adjustment process. The main benefit of adopting a neural network is that it is considerably an optimal solution for datasets which do not have underlying Gaussian distribution. The adaptation mechanism is also in an additive form the same as in the *GA* technique. The formula of the *NNet* technique is depicted as:

$$Effort(P_{new}) = \frac{1}{k} \sum_{i=1}^k (Effort(P_{analogue_i}) + f(P_{new}, P_{analogue_i})), \quad (7)$$

where  $f(P_{new}, P_{analogue_i})$  is an output value from the neural network model, trained to tell the effort difference of a pair of project cases, given their difference in project features. The main difficulty in adopting this adaptation technique is due to a very large configuration possibility for its optimization. This possibly requires many parameter optimizations such as a number of hidden layers and learning procedures. Therefore, neural network approaches in many studies cannot be reproduced. For example, while many replicated results from the Azzeh study (Azzeh 2012) were in agreement with the other proposed studies, the results based on neural networks contradicted the results of Li et al. (2009).

### 2.3 Assessing a Stable Ranking of Estimation Methods

A few years earlier, conflict results regarding comparisons of multiple effort estimation methods (a.k.a. ranking instability) were an issue subject to debate in software effort estimation research (Kocaguneli et al. 2012c), and the answer to questions which often came up such as “Which is the best estimator?” were not easily addressed. This debate existed because if a researcher or an industrial practitioner read through the literature of available methods, divergent conclusions regarding the performance or superiority of a selected method could be found. For examples (Li et al. 2009) reported that the solution adaptation based on the *NNet* technique was accurate, whereas its replication by Azzeh (2012) was not.

Most recently, Keung et al. (2013) revisited the ranking instability issue in effort estimation methods following the guideline suggested in Menzies et al. (2010). They proposed a Stable ranking indication method for model-based effort predictors, and also proved that their indication method provided a stable ranking list of 90 variants of effort predictors, which commonly appeared in the effort estimation literature. According to Keung et al. (2013), 3 conditions should be carefully controlled for a stable ranking result when undertaking experiments to compare multiple methods:

- Variants of dataset instances are sufficient to draw a conclusion;
- The procedure to sample the training/test instances is logical and replicable;
- The performance evaluation criteria are sufficient in amount and are valid.

In the study of Keung et al. (2013), the leave-one-out approach was selected as the sampling method, because it does not rely on a random selection of training/test split (Kohavi

1995). The validity of the performance criteria was maximized by a means of 7 error measures, and the method for summarizing the 7 measures was a robust statistical method, namely, *win-tie-loss* statistics (Demšar 2006). Also, for the generalization of the results, they applied 20 datasets, together with different model design options to draw a stable conclusion.

### 3 Methodologies

In order to generate the stable ranking list of solution adaptation techniques, we adopted and extended the Stable ranking indication method proposed by Keung et al. (2013). The aim of the extension to this method is validity improvement. This section explains procedures we adopted and extended in detail.

All the adaptation techniques we evaluated in this study are implemented in the solution adaptation stage of the basic ABE0-kNN framework (Keung 2009) with the *best-k* selected in every single test (Baker 2007). We denote the name of the framework as ABE0-*best-k*NN for reference later in this paper. All the implementations were developed in MATLAB and were executed on a standard Intel Core i7 notebook with 16 GB of main memory. For compute-intensive experiments such as the adaptation techniques based on neural networks, we tailored these experiments with General-purpose computing on graphics processing units (GPGPU) technology to accelerate the experiment, as suggested by Phannachitta et al. in their proposal of the *ABE-CUDA* framework (Phannachitta et al. 2013).

Given that there are 8 adaptation techniques in combination with 4 feature subset selection methods to evaluate in this study, we generated 32 variants of configurations by implementing all the pairing between adaptation techniques and feature subset selection methods in the ABE0-*best-k*NN framework, i.e., feature selections were applied in the project retrieval stage, and adaptation techniques were adopted in solution adaptation stage. Keung et al. (2013) suggested the need to examine different feature subsets in experiments because the same estimation model often shows varying performance on different feature subsets selected prior to the model building.

#### 3.1 Dataset Instances

We selected 12 real world software project datasets in our experiments. All the datasets were available in the tera-PROMISE software engineering repository (accessible through (Menzies et al. 2015)), where datasets are made available and commonly reused in software development effort estimation studies. As listed and detailed in Table 2, the selected 12 datasets consisting of 951 software projects are diversified in software size and project characteristics.

For a brief explanation for each selected dataset, Albrecht contains software projects from IBM completed in 70s (Albrecht and Gaffney 1983). China contains software projects from multiple Chinese companies (Menzies et al. 2015). Project cases in Cocomo-sdr were from various companies in Turkey (Bakır et al. 2010). The Cocomo81 dataset (Boehm 1981) contains NASA software projects. Desharnais is a well-known dataset containing software project developed in Canada (Shepperd and Schofield 1997). Finnish contains project cases from different companies in Finland (Kitchenham and Känsälä 1993). Kemerer is a small dataset from large business (Kemerer 1987). Maxwell contains project cases from commercial banks in Finland (Maxwell 2002). Miyazaki94 contains many software projects developed in COBOL language (Miyazaki et al. 1994). Nasa93-c1, Nasa93-c2, and

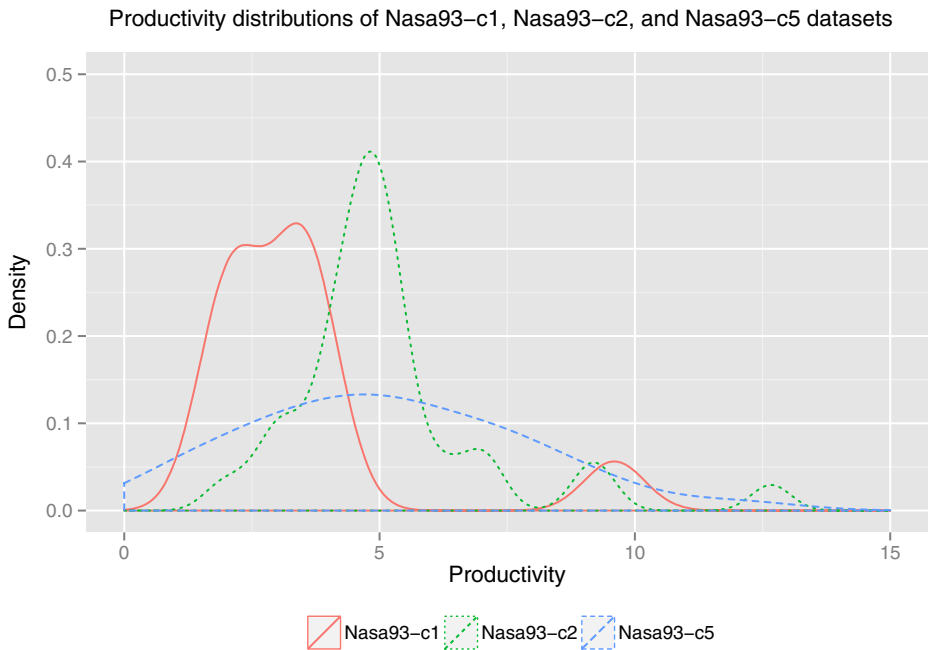
**Table 2** The 12 selected datasets consisting of 951 software project cases ( $F$  = number of features,  $F_{Cat}$  = number of categorical variables,  $SS$  = number of software size variables,  $N$  = number of projects)

Datasets	F	$F_{Cat}$	SS	N	The effort variables				
					Unit	Min	Mean	Median	Max
Albrecht	7	0	2	24	months	0.5	21.9	11.5	105.2
China	18	0	1	499	hours	26.0	3921.1	1829.0	54620.0
Cocomo-sdr	22	0	1	24	months	2.0	32.0	12.0	342.0
Cocomo81	17	1	1	63	months	5.9	683.5	98.0	11400.0
Desharnais	11	1	2	81	hours	546.0	4833.9	3542.0	23940.0
Finnish	8	2	1	38	hours	460.0	7678.3	5430.0	26670.0
Kemerer	7	2	2	15	months	23.2	219.2	130.3	1107.0
Maxwell	27	6	1	62	hours	583.0	8223	5190.0	63694.0
Miyazaki94	8	0	1	48	months	5.6	87.5	38.1	1586.0
Nasa93-c1	16	4	1	12	months	24.0	139.9	66.0	360.0
Nasa93-c2	16	4	1	37	months	8.4	222.9	82.0	1350.0
Nasa93-c5	16	4	1	48	months	72.0	1011.0	571.4	8211.0
			Total	951					

Nasa93-c5 are homogenized datasets being decomposed from the Nasa93 dataset. The postfix c-1, c-2, and c-5 indicated the development centers where each software project was developed.

Out of the 12 datasets, 3 datasets (Nasa93-c1, Nasa93-c2, and Nasa93-c5) are homogenized datasets. In the literature (Keung et al. 2013; Kocaguneli et al. 2012a, b), dataset decomposition is a technique commonly applied to the Cocomo81, Desharnais, and Nasa93 datasets before performing an experiment based on them. The common policy used to decompose these datasets is to divide project cases into separated datasets based on a categorical variable that clearly distinguishes the cases with the same attribute values from the others. This policy is the same as that commonly used in providing categorical variables for the *RTM* adaptation technique. Specifically, the Desharnais dataset is commonly divided by an “Environment” variable into 3 subsets called Desharnais-L1, Desharnais-L2, and Desharnais-L3. The Cocomo81 dataset is commonly decomposed into 3 datasets named Cocomo81-e, Cocomo81-s, and Cocomo81-o using its “Mode” variable. For the Nasa93 dataset, the “Development center” was the variable of concern, which is commonly used to divide it into Nasa93-c1, Nasa93-c2, and Nasa93-c5 datasets.

However, one possible major concern is that the use of some decomposed datasets may be questionable unless it can be confirmed that the distributions of all those decomposed from the same datasets should be treated as independent. In contrast to the previous studies (Keung et al. 2013; Kocaguneli et al. 2012a, b), we question that decomposing the Desharnais and Cocomo81 datasets may not be entirely valid. This is because, for example, the Desharnais-L1 and Desharnais-L2 datasets being decomposed using the “Environment” variable are divided into sub-datasets that were developed using Basic Cobol and Advanced Cobol languages, respectively, while there seems to be no available information to assess how is the difference between the two development languages. Similarly, decomposing the Cocomo81 dataset by using the “Mode” variable would also lead to questionable sub-datasets. That is, all the development languages and productivity are greatly overlapped



**Fig. 2** Kernel distribution plots of effort and productivity of the Nasa93-c1, Nasa93-c2, and Nasa93-c5 datasets

across all the datasets being decomposed from the Cocomo81. In our view, only the datasets decomposed from Nasa93 appear to be valid as they are decomposed with regards to the development centers. Nonetheless, their empirical distributions may still need to be confirmed as being different.

As suggested by Kitchenham (2015), we used Kernel distribution plots to show the summary of the distributions of the Nasa93-c1, Nasa93-c2, and Nasa93-c5 datasets. Due to the problem of space, we only show the Kernel distribution plots of productivity (i.e., effort / software size) of all these 3 datasets in Fig. 2. In this figure, the Kernel distribution plots show that the productivity distributions of the Nasa93-c1, Nasa93-c2, and Nasa93-c5 are different. Hence, in our experiments of this study, we decided to use Nasa93-c1, Nasa93-c2, and Nasa93-c5 as decomposed datasets, while treating Cocomo81 and Desharnais as full datasets.

### 3.2 Four Feature Subset Selection Methods Adopted

Feature subset selection is one of the most important steps in the performance of analogy-based estimation (Chen et al. 2005; Keung et al. 2013). We argue that a comparison of the test instances between adopting of one single feature selection method versus adopting no feature selection as in Azzeh (2012) may not be sufficient to draw a stable conclusion because a method that relies on strong correlation between features and effort may bias a feature selection criteria towards linear models. In our evaluation, we adopt 4 feature selection methods in our experiments, which have been commonly applied in effort estimation literature (Kocaguneli et al. 2012c; Keung et al. 2013) namely:

**All** selects all the features in a dataset;

**Sfs** (Sequential forward selection (Kittler 1986)): is a greedy algorithm that adds features into an initially empty set one by one until there is no further improvement when adding any of the other remaining features;

**Swvs** (Stepwise variable selection (Kittler 1986)): iteratively adds or removes features from the feature set based on the statistical significance of a multilinear model in explaining the objective variable. In each iteration it builds two models that include and exclude one selected feature. The process is repeated until there is no additional improvement by either adding or removing a feature from the list of selected features. The criteria we used to decide whether to add or remove a feature is the same as used in stepwise regression, which is the F-statistic significance of a regression model (Alpaydin 2014). The estimated target value from *Swvs* is the regression result subject to the list of features, selected by the last step of feature selection process;

**Pca** (Principal component analysis (Li et al. 2007; Wen et al. 2009; Tosun et al. 2009)): can be used as a feature selection method since its main purpose in the machine learning domain is to reduce the feature dimension. *Pca* uses a transformation function to map the entire dataset into a new lower-dimensional space, where remaining values are uncorrelated and retained the essential variance of the original dataset.

### 3.3 Procedure to Sample the Training and Test Instances

We sampled training and test instances by using leave-one-out approach (Kohavi 1995). Given  $N$  project cases, 1 project at a time becomes a test instance of the model that is trained from all the remaining  $N-1$  cases. The leave-one-out approach performed over an  $N$ -case dataset has  $N$  predictions, and all these  $N$  predictions will hold the same values in every single run because the leave-one-out does not depend on a random sampling of the cases. Leave-one-out is our choice of sampling procedure because leave-one-out has proved to be the procedure that generates the lower estimation bias, and is more robust when experimenting on small and medium-sized datasets (Kocaguneli and Menzies 2013; Kosti et al. 2012) (i.e. less than 100 samples), where most of the available effort datasets are within this range (Menzies et al. 2015).

### 3.4 Multiple Performance Measures (Evaluation Criteria)

Two types of performance measures were used in our experiment: (1) error measures and (2) a statistical test method to summarize the error measures. Our choice of error measures follows the suggestion by Foss et al. (2003), who performed a prominent statistical analysis and proved that many performance measures used as accuracy indicators in effort estimation, such as the well-known *MMRE* and *Pred(25)*, are biased and are no longer believed to be trustworthy measures. Therefore, our list of error measures is different from that which commonly seen in the literature including that of Azzeh (2012) and Keung et al. (2013), because we select 5 error measures from the list of more stable measures suggested by Foss et al. (2003). A detailed explanation of the 5 error measures is as follows:

The absolute residual ( $AR$ ) is the absolute difference between the predicted and the actual effort value. The overall average error  $AR$  is commonly derived as the Mean Absolute Error ( $MAR$ ), and Median Absolute Error ( $MdAR$ ):

$$MAR = \text{mean}(\text{all}|Actual_i - Predicted_i|); \quad (8)$$

$$MdAR = \text{median}(\text{all}|Actual_i - Predicted_i|), \quad (9)$$

where  $Actual_i$ , and  $Predict_i$  are the actual and the predicted effort of a software project case  $i$ .

The other 3 error measures are variants of standard deviation, which are simple measures of residual error: Standard Deviation ( $SD$ ), Relative Standard Deviation ( $RSD$ ), and Logarithmic Standard Deviation ( $LSD$ ), and they are computed as follows:

$$SD = \sqrt{\frac{\sum_{i=1}^N (Actual_i - Predicted_i)^2}{N - 1}}; \quad (10)$$

$$RSD = \sqrt{\frac{\sum_{i=1}^N \left(\frac{Actual_i - Predicted_i}{Size_i}\right)^2}{N - 1}}; \quad (11)$$

$$LSD = \sqrt{\frac{\sum_{i=1}^N \left(e_i - \left(-\frac{s^2}{2}\right)\right)^2}{N - 1}}, \quad (12)$$

where  $Size_i$  is a single software size variable (e.g., adjusted Function points),  $e_i$  is given by  $\ln(Actual_i) - \ln(Predict_i)$ , and  $s$  is an estimator of the variance of  $e_i$ ). As noted in the proposed study by Foss et al. (2003), the mean and the variance of errors of a model being taken by logarithms will be equal to  $-s^2/2$  and  $s^2$ , respectively, if they exhibit normal distribution. Hence, in other word, (12) indicates a simple standard deviation of errors being normalized to logarithmic scale.

After the leave-one-out experiment produced  $N$  predictions, the estimation performance was recorded by the 5 error measures subject to a statistical test based on Brunner test (Brunner et al. 2002). The Brunner test can be considered as one of the most robust and reliable variants of the Wilcoxon rank-sum test (Demšar 2006), a commonly adopted non-parametric test in empirical software engineering studies (Kitchenham 2015; Kocaguneli et al. 2012a, b; Keung et al. 2013) that is practically useful when the samples do not underly normal distribution. Given 32 variants of configurations of analogy-based estimators (i.e. ABE0-best-kNN being tailored with 8 adaptation techniques and 4 feature selection methods), the Brunner test collected the performance measures of 32 configuration variants of ABE0-best-kNN estimators such that each was compared with the 31 others. A statistical measure that states one variant is better, on par, or worse than how many other variants is called *win-tie-loss* statistics (Azzeh 2012; Keung et al. 2013; Kocaguneli et al. 2012c, 2013a). We used this statistical method to summarize the overall performance for each predictor, and this overall performance was then used to generate the stable ranking list of the 32 ABE predictors.

The test procedure for comparing a pair of ABE0-best-kNN variants  $i$  and  $j$  is depicted in Fig. 3. When the Brunner test showed that the estimated effort from any of the two ABE variants were not significantly different (i.e.,  $p\text{-value} > 0.05$ ), we recorded no significant difference between the two techniques (i.e., they are tied). Otherwise, the technique with the lower error value will win the comparison. Note that for all the 5 error measures we used, a lower number indicates a better performance.

```

1: if Brunner test( $Err_i, Err_j, 0.05$ ) says they are the same then
2:    $tie_i = tie_i + 1$ 
3:    $tie_j = tie_j + 1$ 
4: else
5:   if better( $Err_i, Err_j$ ) then
6:      $win_i = win_i + 1$ 
7:      $loss_j = loss_j + 1$ 
8:   else
9:      $win_j = win_j + 1$ 
10:     $loss_i = loss_i + 1$ 
11:   end if
12: end if

```

**Fig. 3** Comparing the performance between estimator  $i$  and  $j$ , on their single error measure  $Err_i$  and  $Err_j$ . Lower values indicate better performance for all 5 error measures

The main reason that we selected the Brunner test as our main statistical test method of this study was that, as suggested in a simulation study by Zimmerman (2000), unequal variance can greatly reduce the power of any test based on the Wilcoxon test, even if the two test groups have equal sample sizes. To avoid such invalid statistical inferences, we followed the guidelines provided by Wilcox (2011) and applied the Brunner test, which can be considered as the Wilcoxon test based on the confidence interval of the  $p$ -hat metric. This  $p$ -hat metric considers a probability of random observations instead of a direct ranking comparison as done by the conventional Wilcoxon test. The direct ranking comparison was considered the main causes of problem, pointed out by (Zimmerman 2000). Overall, the use of the  $p$ -hat metric contributes to the Brunner test as to improve the statistical power, reduce the risk of committing the type I error, and make the test better prone to the effect of outliers (Kitchenham 2015; Wilcox 2011), compared with the conventional Wilcoxon rank-sum test. These were otherwise especially problematic when the samples' size (e.g., the size of data points) are small, where most of the datasets available in the tera-PROMISE repository are this size (Menziés et al. 2015). We therefore believe that using multiple stable error measures and a valid statistical test method will enable replicability, and at the same time, will maximize the validity of the experimental results.

For the implementation of the Brunner test in our experiments, we used a function named *bprm*, being made available by Wilcox (2011) in a robust statistical test library named *WRS* in R language system.

### 3.4.1 Interpretation of the Win-Tie-Loss Statistics

Based on the win-tie-loss statistics, the performance of effort estimators being evaluated is commonly assessed by means of ranking in regard to the total counts of *wins*, *losses*, or *wins - losses*. The rank of an effort estimator is based on how many times an estimator significantly outperforms the others or being outperformed by the others, where the level of significance is indicated by the Brunner test. According to Demšar (2006) and Kocaguneli et al. (2012c), there is no generally accepted method to compare these summations. In this study we interpreted *wins - losses* to generalize the experimental results. And for more specific results, we interpreted all of them to observe stability through the agreement established across different rankings.

The procedure in Fig. 3 was executed over all pairing of configurations between adaptation techniques and feature selection methods across all error measures. After a single execution, the summations of *wins*, *ties*, and *losses* for each design variant held the values that indicated its estimation performance. Then, we interpreted the summations in the following 3 ways:

1. We calculated *wins - losses* and ranked the 32 ABE variants based on *win-tie-loss* statistics. In this way, we could clearly indicate the poorly performed predictors which have *wins - losses* less than 0 (i.e., the number of *wins* is less than the *losses*). Furthermore, according to our empirical results, the ranking based on *wins - losses* is in broad agreement with most rankings computed separately for each error measure over *wins*, *losses*, and *wins - losses*.
2. We calculated the mean rank change, which indicates the agreement among each single error measure across different summation procedures of the *win-tie-loss* statistics. The calculations are:
  - (a) Calculate *wins*, *loss*, and *wins - losses* across all error measures;
  - (b) Calculate *wins*, *loss*, and *wins - losses* separately for each error measure, each of which has its own ordering of the 32 configuration variants;
  - (c) Calculate the absolute difference in ranking between (a) and (b) for each configuration variant;
  - (d) The indicator for stable ranking is the mean rank changes derived from 5 error measures  $\times$  3 statistics (*wins*, *loss*, and *wins - losses*).
3. From 2(b), we calculated the minimum rank, averaged rank, and maximum rank for each of the 5 (error measures)  $\times$  3 (statistics) = 15 ranking lists, and averaged them across the 5 error measures. These indicators can ensure stability in further detail which is more specific than the summarized rank change.

## 4 Results

We applied the leave-one-out sampling approach to 12 datasets and recorded the estimated effort values achieved by 32 configuration variants of the ABE0-*best-kNN* estimator. Table 3 presents the ranking list of the 32 configuration variants of ABE0-*best-kNN* that were cross-generated from 8 solution adaptation techniques and 4 feature subset selection methods. The maximum number of *wins - losses* for a single configuration variant is 31 comparisons  $\times$  5 error measures  $\times$  12 datasets = 1,860 (i.e. all wins and no losses). The result in Table 3 is ordered by the total number of *wins - losses* accumulated from all comparisons.

### 4.1 RQ1 How Stable is the Ranking List We Discovered?

The column indicating the rank change in Table 3 shows that we are able to determine stable ranks based on this ranking list of configuration variants of the ABE0-*best-kNN* estimator. The majority of observations (over 70 %) had mean rank change  $\leq 6$  (less than 20 % of the maximum possible changes). Therefore, this stable ranking list is leveraged by the robust statistical method and the robust error measures used in our study to provide overall stability.

In addition to the low average rank change observed across Table 3, we see that the mean average rank, which was averaged by 15 ranking lists (from 5 error measures  $\times$  3 summation



**Table 3** Win-tie-loss ranking results from the experiments

Rank	wins- losses	Solution adaptation technique	Feature selection method	Mean			
				rank change	minimum rank	maximum rank	average rank
1	449	RTM	Pca	4	2.4	8.6	5.1
2	428	RTM	Sfs	3	2.2	8.2	5.2
3	373	RTM	All	5	3.6	11.6	7.3
4	336	RTM	Swvs	5	7.2	9.8	8.5
5	325	LSA	All	3	5.8	10.2	7.7
6	278	MSA	All	4	8.8	12.4	10.3
7	275	UAVG	Swvs	4	8.0	10.0	8.9
8	264	LSA	Sfs	5	8.4	14.0	10.7
9	262	MSA	Swvs	6	8.8	14.8	11.2
10	252	MSA	Pca	3	5.4	12.4	9.3
11	232	MSA	Sfs	6	10.2	17.0	12.9
12	220	LSA	Swvs	4	10.2	15.0	12.1
13	180	LSA	Pca	3	10.0	15.4	12.8
14	141	GA	Swvs	3	10.0	15.2	13.0
15	96	IRWM	Swvs	4	13.2	18.6	15.7
16	45	NNet	Swvs	7	8.2	24.4	15.9
17	-6	AQUA	Swvs	6	11.4	20.2	15.9
18	-89	UAVG	Sfs	3	17.6	22.4	19.7
19	-138	IRWM	Sfs	3	17.2	23.6	20.3
20	-155	UAVG	All	3	20.6	24.2	22.0
21	-176	UAVG	Pca	3	18.8	23.2	20.7
22	-189	IRWM	All	2	21.8	24.2	22.7
23	-194	IRWM	Pca	6	16.2	25.0	20.7
24	-250	AQUA	All	8	15.4	23.0	20.1
25	-253	AQUA	Pca	8	13.8	25.4	20.5
26	-279	AQUA	Sfs	9	15.4	23.0	20.3
27	-284	NNet	Pca	4	20.8	25.0	23.3
28	-321	GA	Sfs	6	17.4	30.0	24.3
29	-338	GA	All	6	17.2	31.0	24.9
30	-381	NNet	All	4	23.0	29.6	26.4
31	-508	NNet	Sfs	2	26.8	31.2	29.1
32	-595	GA	Pca	2	28.2	31.4	30.2

Results for each combination of configuration variants of feature selection methods and solution adaption techniques are sorted by *wins - losses*. The column that indicates mean rank change shows how stable a configuration variant is; a lower number represents a more stable rank. The 3 columns to the right indicate the mean minimum rank, mean maximum rank, and mean average rank, calculated separately for each configuration variant by the 5 error measures (MAR, MdAR, SD, RSD, and LSD)

procedures of *win-tie-loss* statistics) were not much different to the ranking based on the total summation of *wins-losses*. The maximum ranking difference between the ranking in

Table 3 and the mean average rank was only 5.7 (at rank #26), and more than half of the configuration variants of the analogy-based estimators had such difference values less than 2. This observation ensures the overall stability of the ranking methodology we used and the ranking list we discovered.

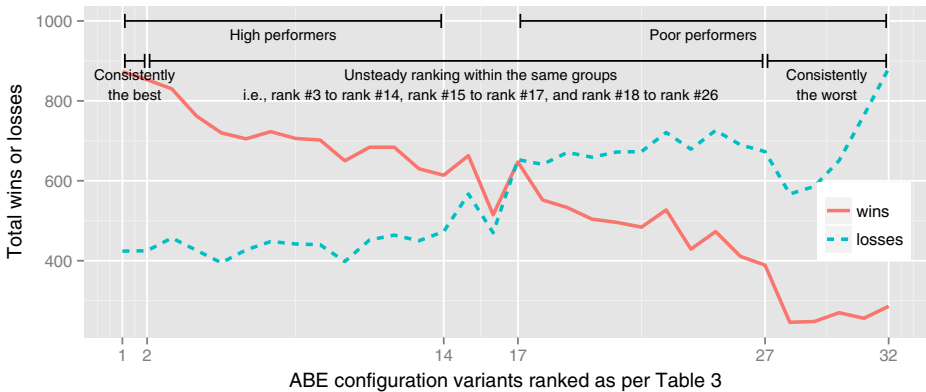
Since our results were observed with many configuration variants between solution adaption techniques and feature selection methods, and some solution adaption techniques are based on similar adaptation mechanism, these configuration variants may thus have a very similar accuracy performance. For example, *UAVG* and *IRWM* do not apply any descriptive variables to adjust the effort, thus, we expect more *tie* results observed between pairs of configuration variants with a similar adaptation mechanism. In our motivated study of Keung et al. (2013), a massive change of ranking in some portions of their ranking list of effort predictors could imply a situation in which all the predictors appearing in the same portions of the unsteady ranking list would have a similar performance. However, as we increased the robustness of the statistical method and of the error measures, the ranking based on our extended method rarely changed across different measures. We therefore extended the Keung et al.'s Stable ranking indication method to additionally observe the mean minimum rank, mean maximum rank, and mean average rank, and we compared them all with the mean rank change. Results based on an this extension show that we can now observe the configuration variants in Table 3 which have a similar accuracy performance.

Observing the agreement of the ranking between the ranking based on the summation by *wins - losses* and by the mean average rank, we can divide Table 3 into 5 portions based on performance and the stability. We see that the top 2 ranks were very stable and consistently outperformed. In the second group from rank #3 to rank #14, we see that all these configuration variants may have the same performance, as their mean rank change, mean maximum rank, and mean average rank were not much different ( $\mu_{\text{mean average rank}} = 10.4$  and  $\sigma_{\text{mean average rank}}^2 = 2.05$ ). In the next portion of the ranking list between rank #15 and rank #17, all the configuration variants in this portion consistently had the same performance, as their mean average ranks were rounded to rank #16. After rank #16, the ranking became interchangeable until rank #26. Similar to the second group, the configuration variants between rank #18 to rank #26 also have very similar performance, as their mean maximum rank and mean average rank were almost equal ( $\mu_{\text{mean average rank}} = 20.8$  and  $\sigma_{\text{mean average rank}}^2 = 1.00$ ). In summary, our ranking list of 32 configuration variants of the *ABE0-best-kNN* estimator based on the extension to the Stable ranking indication method by Keung et al. (2013) is stable. Also, when more indicators are used (e.g. mean average rank), we can determine which portions of the stable ranking list are more stable than the other portions.

## 4.2 RQ2 Which are the High-Performance Solution Adaption Techniques for Analogy-Based Estimation?

In the study of Keung et al. (2013), the use of cut off between the portions of the ranking list with a clearly different amount of rank changes could identify the number high-performance effort learners. The authors indicated the stable-ranked effort learners on the top ranks above the cut off as the learners with a high performance and high rank. We applied the same idea of Keung et al. here, but we used the cut off criteria we explained in RQ1 since our ranking list is too stable to apply the cut off by using only rank change.

In Table 3 the values of *wins-losses* and the mean average rank show a notable cut off for the high-performance and high-ranked variants of the *ABE0-best-kNN* estimator at the rank below #15. The configuration variants on the top 14 ranks had their *wins-losses* higher



**Fig. 4** The sum of win and loss values for all configuration variants of the ABE0-*best-k*NN (over all error measures and all datasets). The maximum value of *wins* and *losses* is 31 comparison per one variant  $\times$  5 error measures  $\times$  12 datasets = 1,860

than 141 and their mean average ranks less than 13 when evaluated by any other means of *win-tie-loss* statistics. The results suggest that these 14 configuration variants of the ABE0-*best-k*NN estimator are the high-performers.

Table 3 shows that 12 out of the 14 configuration variants of the ABE0-*best-k*NN, they were based on all the possible combinations between the *RTM*, *LSA*, and *MSA* techniques and the 4 feature subset selection methods selected in this study. This thus narrows down the set of high-performance solution adaptation techniques into a set of 3 techniques: *RTM*, *LSA*, and *MSA*. The characteristics of these 3 techniques are distinctive from the others in that they are based on a linear extrapolation between software size variable(s) and the effort required. We therefore endorse the solution adaptation techniques with this adaptation mechanism as a good bet for good performance when applied to an analogy-based estimator.

Since we see the overall stability of the ranking in Table 3 was very high, and the mean minimum rank, the mean maximum rank, and the mean average rank could identify the cut off to divide the ranking list by performance, Fig. 4 was produced as a summary of our results of Table 3. The x axis of Fig. 4 represents the order of the 32 variants of ABE0-*best-k*NN configured with 8 solution adaption techniques together with 4 feature selection methods. The x axis is sorted by the ranking of Fig. 3 (by *wins - losses* in descending order). The y axis shows the number of *wins* and *losses* of the 32 variants of ABE0-*best-k*NN regarding the sorted order by *wins - losses*.

A summary of the performance and stability ranking of the 8 solution adaptation techniques configured with 4 different feature selection methods is listed as follows:

1. The pairs of (solution adaptation technique, feature selection method) configured to the ABE0-*best-k*NN appearing on the top 14 ranks had a *high performance*;
  - (a) The ranking of the top 2 variants were very stable;
  - (b) The variants in the ranking from rank #3 to rank #14 were very similar in terms of performance;
2. ABE0-*best-k*NN variants in the ranking from rank #15 to rank #17 showed *moderate performance*, and their performance were almost the same;
3. ABE0-*best-k*NN variants in the ranking from rank #18 to rank #26 were considerably stable across their ranking, but they had relatively *poor performance*;

**Table 4** The average wins, losses, and wins-losses results that were aggregated from 4 feature selection methods

Adaptation technique	Average	Average	Average	Group by	
	Wins	Losses	Wins - Losses	Performance	Mechanism
RTM	829.50	433.00	396.50	high	type 1
MSA	685.25	429.25	256.00		
LSA	685.00	437.75	247.25		
UAVG	568.75	605.00	−36.25	moderate	type 2
IRWM	551.75	658.00	−106.25		
AQUA	490.00	687.00	−197.00	poor	type 3
GA	348.50	626.75	−278.25		
NNet	357.50	639.50	−282.00		

This ranking list from the summation procedures based on *wins* and *wins-losses* are in broad agreement

#### 4. The bottom 6 variants had consistently *poor performance*.

Note that we listed the variant configured with the *AQUA* adaptation technique and having the feature subset selected by the *Swvs* method (rank #16) in the second group even if it had *wins - losses* less than 0, because its mean average rank was rounded to its exact rank of #16 and its mean minimum rank had almost reached rank #11.

### 4.3 RQ3 What is the Overall Performance of each Adaptation Technique?

To summarize the overall performance of each adaptation technique, we calculated the average of *wins*, *losses*, and *wins - losses* separately over each of the 4 feature selection methods. The results are presented in Table 4, showing that the rankings by average *wins* and average *wins - losses* were almost identical, with only the ranks difference between the *GA* and *NNet* techniques. When we look closely at Table 4, we see that the average of *wins* and *wins - losses* can clearly divide the 8 adaptation techniques into 3 ranges based on their accuracy of performance. Furthermore, we observed that the ranking list of the adaptation techniques in Table 4 also divide techniques based on their adaptation mechanism. Recall the mechanism of the 8 solution adaptation techniques in this study as summarized in Table 1, we can group similar adaptation techniques together by their adaptation mechanism and performance as follows:

**Tier 1** - highly performed techniques: *RTM*, *MSA*, and *LSA* are techniques that adjust effort with linear extrapolation between one or a few feature variables to the effort.

**Tier 2** - moderately performed techniques: *UAVG* and *IRWM* are techniques that only aggregate effort from multiple project analogues.

**Tier 3** - poorly performed techniques: *AQUA*, *GA*, and *NNet* are techniques that adjust effort with the entire feature subsets selected by a feature selection method.

Following these mechanisms, we are able to indicate that the estimation performance of the tier 3 techniques has its upper bound below tier 2, which is less desirable in most of the cases. Therefore the ranking list of solution adaption techniques in Table 4 is conclusive and

can be used to derived the speculation that adaptation techniques with similar adaptation mechanisms will produce a similar performance in terms of estimation accuracy.

## 5 Discussions

### 5.1 General Discussion

The results from the comprehensive experimental procedure in our study show that we can overcome the ranking instability issue (Keung et al. 2013) in choosing a solution adaptation technique for analogy-based estimation. According to the study by Keung et al. (2013) that overcame ranking instability in model-based effort predictions, there are many factors to influence the rank changes of the effort estimators which are related to the methodologies that generated the ranking list. The factors can be categorized into two general situations: (1) the use of different methodologies among different literatures (e.g. using different sampling methods to generate the training/test set), and (2) an inappropriate experimental design (e.g. the commonly seen misuse of a biased *MMRE* error measure). Our results show that these two general situations were also the causes of ranking instability in solution adaptation techniques for analogy-based effort estimation.

The stability of our ranking was ensured by a broad agreement across multiple stable measures that were used to assess all the 8 adaptation techniques selected in our study. Furthermore similar behaviors were also observed, such as similar techniques in terms of adaptation mechanism preferred similar feature selection methods and could deliver similar accuracy performance. Thus, we can offer observations based on groups of techniques, in which we can generalize the results of our study to other techniques which have similar adaption mechanisms in this study. The general observations are as follows:

**Finding 1** Solution adaptation techniques that utilized linear adjustment functions are consistently more accurate than those of non-linear techniques. The evidence for this finding is, the *NNet* technique, the only techniques that is based on non-linear techniques, consistently ranked lower than the other 7 adaptation techniques. Nonetheless, any techniques that adopted neural networks as a learning model are often very sensitive to the configuration and setup parameters used. In our study we configured the network with the same configuration parameters taken from the replication study by Azzeh (2012), which were said to be taken exactly from the original proposed approach. Therefore, we believe that our finding based on linear versus non-linear techniques is conclusive.

**Finding 2** The assessment based on the *win-tie-loss* also revealed that adaptation techniques that adjust the effort values using only one or a few features related to software size performed consistently better than techniques that did not apply any features to the effort adjustment, and were also considerably better than techniques that adjusted effort by learning to the difference among the entire set of features. This finding suggests the usefulness of software size features in analogy-based estimation. Although (Kocaguneli et al. 2012b) suggested that estimating an effort dataset with either software size present or absent showed no significant difference in performance, we agree with them that the absence of size features may not influence much in performance only if the size features are not extensively used in the solution adaptation stage. According to our findings, we further suggest that if

the software size features are presented in a dataset and they are also used as adaptation features, then the performance should be significantly improved. This was especially the case in *RTM* or *LSA* as the adaptation techniques.

**Finding 3** We speculate that selection of feature selection methods was important and did influence on the performance of all the adaptation techniques. From the 3 groups of analogy-based estimation variants configured across different solution adaptation techniques and feature selection methods that we divided by performance and stability (i.e. as listed under Section 4.2), we observed that, compared with the other groups, the performance of the configuration variants in tier 1 was highly subject to different feature subset selection methods than others. The *RTM* technique performed best with *Pca* or *Sfs* method, and these two pairs of analogy-based estimation variants were consistently the best performers. Also for *LSA*, the mean average rank indicated that it performed better when software features were selected using the *All* method than using any other methods. In this tier of techniques, only the *MSA* technique showed the least difference between their ranking when software features were selected by each of the different feature subset selection techniques.

On the other hand, *Swvs* was consistently the feature subset selection method of choice for all the other solution adaptation techniques in tier 2 and tier 3. Regarding the configuration variants from rank #3 to rank #17, the *Swvs* generated the best ranks for *UAVG*, *GA*, *IRWM*, *NNet*, and *AQUA* techniques. As seen in Table 3, the *UAVG* and *GA* techniques having feature subsets selected by *Swvs* appeared among high-performance techniques, while all their other variants appeared to be the poorly-performed techniques. Hence, these results suggest that *Swvs* is a feature selection method for adaptation techniques with similar adaptation mechanism as those in tier 2 and tier 3.

## 5.2 Usefulness of ABE Models in Practice

Since ABE is a non-parametric estimation method, there might be situations where it does not perform well in any of its forms. We therefore suggest a method named Analogy-X (Keung et al. 2008) for a practitioner to test whether or not ABE models are suitable for a local dataset, before performing an effort estimation based on it. Leveraged by a technique based on Mantel's correlation resampling test, Analogy-X has been a valid method to decide whether to give up on the procedure and not using any forms of ABE at all in practice. A more detailed explanation and discussion on the Analogy-X method is available in (Keung 2008; Keung and Kitchenham 2008; Keung et al. 2008).

Despite Analogy-X being a feasible method for the purpose, one of its inherent limitation in regard to the use of compute intensive algorithm (i.e., Mantel's correlation resampling test) may hinder its use in resource-limited situations. However, we believe that the situations where ABE does not perform well in any of its forms are unlikely to arise in practice. In our motivating study by Keung et al. (2013), the ABE0 model (i.e., the variant of ABE models commonly referred to as a basic standard ABE model) was ranked #27 out of the 90 variants of effort models commonly adopted in practice. This result shows that despite being one of the simplest variants of ABE models, the performance of ABE0 was not necessarily consistently poor. This ABE0 model was also evaluated in our study. It is the ABE model having feature subset selected by the *All* method and effort adapted with the *UAVG* technique. Our results of Table 3 is in agreement with the study by Keung et al. (2013), indicating that this basic ABE model was not consistently a poor performer. Furthermore, our results of Table 3 could successfully indicate many other variants of ABE models, which performed consistently better than ABE0, especially the variants on the top 14 ranks of

Table 3. Hence, we strongly believe that these high performance variants of ABE models are very less likely to become completely useless in practice.

### 5.3 The 3 Tiers of Adaptation Techniques

#### 5.3.1 The Tier 1 Techniques (MSA, LSA, and RTM)

Although (Azzeh 2012) endorsed the *RTM* technique, Shepperd and Cartwright (Shepperd and Cartwright 2005) opposed it because it was not a technique that perfectly fit with all situations. For example, Shepperd and Cartwright suggested that *RTM* is entirely not a technique of choice when the correlation coefficient between predicted and actual productivity is negative. Our empirical results (in Table 3) are in agreement with Azzeh (2012) showing that *RTM* was a high performer for all the 12 datasets selected in our study. Only *RTM* has reached the consistently best ranks by all means of the values of *wins-losses*, mean minimum rank, mean maximum rank, and mean average rank. We therefore endorse *RTM* as a solution adaptation technique being more accurate than others in a general situation. However, we also agree with Shepperd and Cartwright (Shepperd and Cartwright 2005) that there must be certain situations where *RTM* may not perform well.

In our theoretical view, *RTM* needs improvement in its procedures. One obvious case where *RTM* might not apparently be the technique of choice is when there are factors influencing changes of productivity over time. This is because *RTM* assumes that the underlying mean productivity is itself stable for the entire project sources or within the same group of project sources being divided by a categorical variable. Several studies such as Kitchenham et al. (2002) and Premraj et al. (2005) suggested that, in practice, some factors can influence on the relationship between effort and software size, and make them gradually changed over time. For example, development might become more productive as individual developer become more experienced or development methods are improved. Unfortunately, to the best of our knowledge, there seems to be no existing solution yet to compensate this limitation of *RTM*. In other direction, based on the equation defining *RTM* (4), productivity is the ratio of software size to the effort, so it is not always distributed normally. Therefore, the centre of the productivity distribution may not be an adequate measure to approximate  $M$ . Furthermore, an estimation of  $r$  based on a small number of non-normally distributed values is likely to be extremely error prone. Even if *RTM* appears to be the currently best solution adaptation technique at the time of writing, we strongly suggest to improve these components of the technique. One possibility is to improve the estimation of  $M$  to be based on trimmed mean or Winsorized mean (Wilcox 2011). This will increase the chance of having normally distributed productivity in a transformed space. Other future possibility is to apply an idea of adjusting the software size variables to better measure the productivity. A detailed explanation of an example approach for this purpose was suggested and discussed by Kitchenham and Mendes in (Kitchenham and Mendes 2004). Taken together, even if the present study can successfully conclude *RTM* as the best technique by means of generalized performance produced, there are many more possibilities to improve its performance.

We also strongly believe that an improvement of productivity measurement would also further improve the estimation accuracy of the other two techniques in tier 1. All the three techniques in this tier make use of the software size variable in their adaptation procedures following the triangular relationship between software size, productivity, and effort:  $Effort = Size \times Productivity$  (Boehm 1981). Therefore, we can say that these tier 1 adaptation techniques exploit the productivity in their adaption procedure by calibrating the productivity of the new case before adjust the effort. It thus appears to be that the superiority

of these techniques could be leveraged by a further exploitation of the productivity factor. Therefore, interesting future research for the tier 1 techniques will be to further study the productivity factor and its influence on solution adaptation for analogy-based estimation, which will further improve the achievable accuracy performance.

### 5.3.2 The Tier 2 Techniques (*UAVG* and *IRWM*)

There is almost no significant difference between using an unweighted mean and an inverse-ranked weighted mean to aggregated the effort values from the retrieved similar project cases of the new case. When we look closely at Table 3, only their pairing with *Swvs* feature selection method showed different performance between *UAVG* and *IRWM*. However, overall they were similar in both terms of performance and mechanism. Our view of *UAVG* and *IRWM* is that, the two techniques are based on arithmetic aggregation, there is not much possibility to extend the functions of these two techniques and expect much improvement. Therefore, any other applications based on these two techniques such as building an ensemble by using *UAVG* and *IRWM* as an aggregation function to combine multiple methods may not generate a significantly different performance.

### 5.3.3 The Tier 3 Techniques (*AQUA*, *GA*, and *NNet*)

In general, more sophisticate learning models should perform better than simple heuristics such that adapting to effort by genetic algorithm (*GA*) should perform better than by a simple aggregation or arithmetic mean. However our stable ranking results showed otherwise, especially for all the techniques that adopted machine learning models. We rather suspect that the nature of software project data used for performing effort estimation may not fully suitable with parametric machine learning models. That is, in practice, it is very likely occurred that software project cases having the same characteristic (e.g., software size) may coincidentally have different development effort (Phannachitta et al. 2015). In such cases, it is very difficult for any parametric machine learning model to produce an accurate estimate.

In our opinion, a future possibility to improve solution adaptation techniques based on parametric machine learning models is to better handle these coincided cases. At the same time, the phenomenon where project cases with similar characteristics were completed with diversified amount of effort can also be considered as inconsistent project cases (Bosu and MacDonell 2013), which causes less-desirable estimation accuracy. Even if they are otherwise, we believe that approaches to remove such inconsistent software project cases, such as the TEAK method proposed by Kocaguneli et al. (2012a), are applicable for handling these coincided cases, and would possibly contribute to the accuracy improvement of all the solution adaptation techniques in this tier.

As a result, we suggest assessing the influence of data quality in project datasets prior to model building as a more important future research than exploring the application possibility of other machine learning methods in solution adaption for analogy-based effort estimation.

## 6 Threats to Validity

**Internal Validity** questions whether or not different conclusions can be drawn with regard to the different parameter setups for the estimation. To ensure internal validity, (1) we used



many feature selection methods to avoid bias that some particular model may favor only some specific feature selection methods, and (2) we optimized the number of analogies ( $k$ ) selected for each local training instance in the leave-one-out cross-validation procedure, rather than just statically fixed the  $k$  value as 1, 2, or 3, which we often encountered in effort estimation literature (Azzeh 2012).

There are four main possible issues of internal validity in this study. One issue is the single sampling method we used. The second issue is the choice of optimization parameters for adaptation methods that adopted the learning model. The third issue is in regard to massive aggregation of results from datasets with greatly diversified characteristics. The fourth issue is in regard to the comparison between *MSA* and *LSA* techniques. In this study, we choose the leave-one-out as the sampling approach to generate training/test instances, over the other widespread sampling approach, *N*-way cross validation. Following Hastie et al. (2009) and Kocaguneli et al. (2012c), we chose the leave-one-out approach over the other because the leave-one-out generates lower bias estimates and higher variance estimates than the cross validation. Most importantly, leave-one-out is a deterministic algorithm that enables reproducibility for future replication. For a choice of optimization parameters for adaptation techniques based on a learning model, we used the same configuration parameters taken from the replication study by Azzeh (2012) which were said to be taken exactly from the original proposed approaches. However, these configurations may not be yet optimal, as shown by the performance achieved. Hence, we suggest revisiting our stable ranking of solution adaption method methods when the real optimal configure parameters of the *GA* and the *NNet* techniques can be determined. For the third question, even if the use of greatly diversified datasets may not be sound to fully consider the evaluation based on them as *well controlled multiple-comparison tests*, this point appears to be an inevitable validity threat of software effort estimation studies, where a lack of datasets and scarcity of data points have continually been one of main challenges in this field (Kocaguneli et al. 2013a). The limitation is that we cannot design or control the datasets because they already exists as standard benchmark datasets. Nonetheless, we truly believe that the use of the statistical test method of our choice, i.e., the aggregation based on the win-tie-loss statistics subject to the Brunner test (Brunner et al. 2002), can provide sufficiently trustworthy results, where the present study shows empirical evidences that stable conclusions can be drawn based on them. Furthermore, many recent research studies which analyzed the agreement in pairwise comparisons based on win-tie-loss statistics have continually produced important research conclusions for the communities of software effort estimation, even if the procedure of the statistics appear to be very simple (Kocaguneli et al. 2010, 2012a, b, c). For example, Kocaguneli et al. (Kocaguneli et al. 2012c), in their experiments, analyzed both performance and ranking agreements and were able to overcome a long-standing debate in the communities upon the useful of ensemble of effort estimators.

Regarding the comparison between the *MSA* and *LSA* techniques, Table 2 shows that the majority of datasets only have one single software size variable. This means that *MSA* acted the same as *LSA* in the majority of the experiments of this study. In our view, even if in most cases the *win-tie-loss* statistics exhibited *tie* results when comparing between *MSA* and *LSA*, we see that the overall ranking results were sufficiently valid. This is because, in cases a technique won against either *MSA* and *LSA*, it also won against the other. Empirically, the results of Table 4 showed that the ranking of these two methods was consistent. That is, *MSA* or *LSA* were consistently worse than *RTM*, while they performed consistently better than all other techniques. However, future studies are still required to conclude whether one of the methods (*MSA* and *LSA*) is better than the other. For the present study, the results

based on the 12 selected datasets suggested that there seems to be no significant difference between the *MSA* and *LSA* techniques.

**External Validity** questions the generalization of the results. Many findings in this study were explained in a general manner. For example, we generalized and observed the *MSA*, *LSA*, and *RTM* techniques as adaptation techniques of choice that linearly adjusted the effort by using only a few features related to size. The external validity of this study has been maintained by the use of a large number of datasets. According to our reading of the literature in solution adaptation for analogy-based estimation, the number of 12 datasets consisting of 951 project cases are among the largest samples used in effort estimation studies so far. Table 2 shows that the datasets we used were varied in both size and characteristics, such as their origins, their distribution of the effort variables, and the time period in which they were developed. We believe that a diversity of datasets being can produce generalized findings.

**Construct Validity** questions whether or not we are measuring what we intend to measure. In this study our methodology to generate the stable ranking list of solution adaptation techniques has extended the Stable ranking indication method, proposed by Keung et al. (2013). We made two extensions to the method: (1) we replaced the biased and deprecated error measures such as *MMRE* with more statistically stable error measures for the performance evaluation, and (2) our statistical test method used a more robust test function based on the confidence interval given by Brunner et al.'s *p-hat* metric (Brunner et al. 2002). Leveraged by the robust methodology adopted, we presented a stable ranking list of solution adaptation techniques. Furthermore, the evidence for the stability of the ranking list is that the results are based on all the interpretation of all the error measures and the statistical methods provided are in broad agreement.

## 7 Conclusion

Motivated by instability performance results observed across different literatures of solution adaptation techniques for effort estimation based on case-based reasoning (a.k.a. analogy-based estimation), in this study we revisited 8 commonly adopted solution adaptation techniques. Compared with all existing assessments of solution adaptation techniques, our replication is performed on a more comprehensive experimental setup by using a greater number (12) of datasets observed with many (4) feature selection methods, a greater number (5) of performance measures (*MAR*, *MdAR*, *SD*, *LSD*, and *RSD* (Foss et al. 2003)) which were all proved as stable measures, and more robust statistical methods (Brunner test, one of the most robust and reliable alternative non-parametric test to the Wilcoxon rank-sum test (Brunner et al. 2002)). We setup the analogy-based estimation framework with the best-practice configuration, applied leave-one-out (Keung et al. 2013) to generate training/test instances, and tuned the optimal number of neighbors (*best-k*) for every training/test instance (Kocaguneli and Menzies 2013; Kosti et al. 2012). Based on this comprehensive evaluation study, the following findings have been concluded:

1. A more comprehensive evaluation method consisting of stable error measures and a robust statistical test method enabled us to assess the stable rankings of solution adaptation techniques commonly adopted for analogy-based effort estimation.

2. Our ranking is stable. This stability was achieved based on the error measures and statistical methods we provided, which are in broad agreement. In addition, we can also observed similar behaviors from techniques with similar adaptation mechanisms.
3. Adaptation techniques that linearly adapted the effort by using a few relevant features based on software size and productivity, such as regression toward the mean (*RTM*) and linear size adaption (*LSA*) were both high performers and have stable ranks consistently.
4. Adaptation techniques that only aggregated the effort from project analogues, such as unweighted average of the analogues (*UAVG*), were moderately useful, but their ranks were not stable.
5. Adaptation techniques that adjusted the effort value by using the entire list of features either by modeling or heuristic, such as adaptation based on neural networks (*NNet*), consistently performed worse than any others.
6. Selecting feature subset using principle component analysis (*Pca*) shows outstanding performance with the *RTM* technique, and the results based on this combination of solution adaptation technique and feature selection method showed the best performance among all the possible combinations.
7. The choice of feature selection method significantly influenced the accuracy performance for the moderately performed and poorly performed solution adaptation techniques, more so than for the highly performed techniques. Choosing the right feature selection method for the on-average poorly performed techniques made them ranked among moderately performed techniques. Furthermore, choosing the right selection method for the moderately performed techniques could make them rank higher, with considerably better stability.

Based on the observations made throughout this study, we believe that an alternative stable ranking list of adaptation techniques may be generated when revisiting this study with impurity-free datasets. Our ranking result suggests that adapting to effort by using fewer features results in higher accuracy than using the entire set of features (as discussed in Section 5). Hence, the ranking of adaptation techniques discovered in our experiments is not only based on their ability to suggest a robust adjustment for the effort values, but also based on their ability to tolerate dataset impurities. For future research in solution adaptation for analogy-based effort estimation, studying how data quality influences the performance of each individual solution adaptation technique is more important than improving the performance of one single technique or proposing a yet another new technique.

**Acknowledgments** This research was supported by JSPS KAKENHI Grant number 26330086, was conducted as a part of the JSPS Program for Advancing Strategic International Networks to Accelerate the Circulation of Talented Researchers, and was supported in part by the City University of Hong Kong research fund (Project number 7200354, 7004222, and 7004474).

## References

- Albrecht AJ, Gaffney JE (1983) Software function, source lines of code, and development effort prediction: A software science validation. *IEEE Trans Softw Eng* 9(6):639–648
- Alpaydin E (2014) Introduction to machine learning MIT press
- Azzeh M (2012) A replicated assessment and comparison of adaptation techniques for analogy-based effort estimation. *Empirical Softw Eng* 17(1–2):90–127
- Baker DR (2007) A hybrid approach to expert and model based effort estimation. Master's thesis, Lane Department of Computer Science and Electrical Engineering West Virginia University

- Bakır A, Turhan B, Bener AB (2010) A new perspective on data homogeneity in software cost estimation: A study in the embedded systems domain. *Software Qual J* 18(1):57–80
- Boehm BW (1981) *Software Engineering Economics*, 1st edn. Prentice Hall PTR, Upper Saddle River, NJ USA
- Bosu MF, MacDonell SG (2013) A taxonomy of data quality challenges in empirical software engineering. In: *Proceedings of the 2013 Australian Software Engineering Conference*, pp 97–106
- Brunner E, Munzel U, Puri ML (2002) The multivariate nonparametric behrens–fisher problem. *J Stat Plan and Inf* 108(1):37–53
- Chen Z, Menzies T, Port D, Boehm B (2005) Feature subset selection can improve software cost estimation accuracy. *SIGSOFT Softw Eng Notes* 30(4):1–6
- Chiu NH, Huang SJ (2007) The adjusted analogy-based software effort estimation based on similarity distances. *J Syst Softw* 80(4):628–640
- Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. *J Mach Learn Res* 7:1–30
- Foss T, Stensrud E, Kitchenham B, Myrvtveit I (2003) A simulation study of the model evaluation criterion mmre. *IEEE Trans Softw Eng* 29(11):985–995
- Hastie T, Tibshirani R, Friedman J (2009) *The elements of statistical learning: Data mining Inference and Prediction*
- Jørgensen M, Indahl U, Sjøberg D (2003) Software effort estimation by analogy and regression toward the mean. *J Syst Softw* 68(3):253–262
- Kemerer CF (1987) An empirical validation of software cost estimation models. *Commun ACM* 30(5):416–429
- Keung J (2008) Empirical evaluation of analogy-x for software cost estimation. In: *Proceedings of the 2nd ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, pp 294–296
- Keung J (2009) Software development cost estimation using analogy: A review. In: *Proceedings of the 2009 Australian Software Engineering Conference*, pp 327–336
- Keung J, Kitchenham B (2008) Experiments with analogy-x for software cost estimation. In: *Proceeding of the 19th Australasian Software Engineering Conference*, pp 229–238
- Keung J, Kocaguneli E, Menzies T (2013) Finding conclusion stability for selecting the best effort predictor in software effort estimation. *Automated Software Eng* 20(4):543–567
- Keung JW, Kitchenham B, Jeffery DR, et al (2008) Analogy-x: Providing statistical inference to analogy-based software cost estimation. *IEEE Trans Softw Eng* 34(4):471–484
- Kirsopp C, Mendes E, Premraj R, Shepperd M (2003) An empirical analysis of linear adaptation techniques for case-based prediction. In: *Proceedings of the 5th international conference on Case-based reasoning: Research and Development*, pp 231–245
- Kitchenham B (2015) Robust statistical methods: why, what and how: keynote. In: *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*, vol 1
- Kitchenham B, Känsälä K (1993) Inter-item correlations among function points. In: *Proceedings of the 15th International Conference on Software Engineering*, pp 477–480
- Kitchenham B, Lawrence Pflieger S, McColl B, Eagan S (2002) An empirical study of maintenance and development estimation accuracy. *J Syst Softw* 64(1):57–77
- Kitchenham B, Mendes E (2004) Software productivity measurement using multiple size measures. *IEEE Trans Softw Eng* 30(12):1023–1035
- Kitchenham B, Mendes E (2009) Why comparative effort prediction studies may be invalid. In: *Proceedings of the 5th International Conference on Predictor Models in Software Engineering*, p 4
- Kittler J (1986) Feature selection and extraction. *Handbook of pattern recognition and image processing* 59–83
- Kocaguneli E, Gay G, Menzies T, Yang Y, Keung JW (2010) When to use data from other projects for effort estimation. In: *Proceedings of the International Conference on Automated Software Engineering*, pp 321–324
- Kocaguneli E, Menzies T, Bener A, Keung JW (2012a) Exploiting the essential assumptions of analogy-based effort estimation. *IEEE Trans Softw Eng* 38(2):425–438
- Kocaguneli E, Menzies T, Hihn J, Kang BH (2012b) Size doesn't matter?: On the value of software size features for effort estimation. In: *Proceedings of the 8th International Conference on Predictive Models in Software Engineering*. ACM, New York, pp 89–98
- Kocaguneli E, Menzies T, Keung J (2012c) On the value of ensemble effort estimation. *IEEE Trans Softw Eng* 38(6):1403–1416
- Kocaguneli E, Menzies T (2013) Software effort models should be assessed via leave-one-out validation. *J Syst Softw* 86(7):1879–1890
- Kocaguneli E, Menzies T, Keung JW (2013a) Kernel methods for software effort estimation - effects of different kernel functions and bandwidths on estimation accuracy. *Empir Software Eng* 18(1):1–24
- Kohavi R (1995) A study of cross-validation and bootstrap for accuracy estimation and model selection. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pp 1137–1143
- Kosti MV, Mittas N, Angelis L (2012) Alternative methods using similarities in software effort estimation. In: *Proceedings of the 8th International Conference on Predictive Models in Software Engineering*, pp 59–68
- Li J, Ruhe G, Al-Emran A, Richter MM (2007) A flexible method for software effort estimation by analogy. *Empirical Softw Eng* 12(1):65–106

- Li YF, Xie M, Goh TN (2009) A study of the non-linear adjustment for analogy based software cost estimation. *Empirical Softw Eng* 14(6):603–643
- Maxwell K (2002) *Applied Statistics for Software Managers*. Englewood Cliffs, NJ. Prentice-Hall
- Mendes E, Mosley N, Counsell S (2003) A replicated assessment of the use of adaptation rules to improve web cost estimation. In: *Proceedings of the 2003 International Symposium on Empirical Software Engineering*, pp 100–109
- Menzies T, Jalali O, Hihn J, Baker D, Lum K (2010) Stable rankings for different effort models. *Automated Software Eng* 17(4):409–437
- Menzies T, Rees-Jones M, Krishna R, Pape C (2015) Tera-promise: One of the largest repositories of se research data <http://openscience.us/repo/index.html>
- Miyazaki Y, Terakado M, Ozaki K, Nozaki H (1994) Robust regression for developing software estimation models. *J Syst Softw* 27(1):3–16
- Phannachitta P, Keung J, Matsumoto K (2013) An empirical experiment on analogy-based software cost estimation with cuda framework. In: *Proceedings of the 2013 22nd Australian Conference on Software Engineering*, pp 165–174
- Phannachitta P, Monden A, Keung J, Matsumoto K (2015) Case consistency: a necessary data quality property for software engineering data sets. In: *Proceeding of the 19th International Conference on Evaluation and Assessment in Software Engineering*, p 19
- Premraj R, Shepperd M, Kitchenham B, Forselius P (2005) An empirical analysis of software productivity over time. In: *Proceedings of the 11th IEEE International Software Metrics Symposium*, p 37
- Shepperd M, Cartwright M (2005) A replication of the use of regression towards the mean (r2m) as an adjustment to effort estimation models. In: *Proceedings of the 11th IEEE International Software Metrics Symposium*, pp 38–47
- Shepperd M, Schofield C (1997) Estimating software project effort using analogies. *IEEE Trans Softw Eng* 23(11):736–743
- Shepperd M, Kadoda G (2001) Comparing software prediction techniques using simulation. *IEEE Trans Softw Eng* 27(11):1014–1022
- Tosun A, Turhan B, Bener AB (2009) Feature weighting heuristics for analogy-based effort estimation models. *Expert Syst Appl* 36(7):10,325–10,333
- Walkerden F, Jeffery R (1999) An empirical study of analogy-based software effort estimation. *Empirical Softw Eng* 4(2):135–158
- Wen J, Li S, Tang L (2009) Improve analogy-based software effort estimation using principal components analysis and correlation weighting. In: *Proceeding of the 2009 Asia-Pacific Software Engineering Conference*, pp 179–186
- Wilcox R (2011) *Modern statistics for the social and behavioral sciences: A practical introduction* CRC press
- Wilson DR, Martinez TR (1997) Improved heterogeneous distance functions. *J Artif Int Res* 6(1):1–34
- Zimmerman DW (2000) Statistical significance levels of nonparametric tests biased by heterogeneous variances of treatment groups. *J Gen Psychol* 127(4):354–364



**Passakorn Phannachitta** received his B.E. (Hons) in Computer Engineering from Kasetsart University Thailand, and his M.E. and Ph.D. in Information Science from Nara Institute of Science and Technology (NAIST), Japan. He is a postdoctoral researcher in the Graduated School of Information Science at NAIST. His research interests include quantitative methods in empirical software engineering (ESE), software effort and cost estimation, and quality measurement and improvement of ESE datasets.



**Jacky Keung** received his B.Sc. (Hons) in Computer Science from the University of Sydney, and his Ph.D. in Software Engineering from the University of New South Wales, Australia. He is Assistant Professor in the Department of Computer Science, City University of Hong Kong. His main research area is in software effort and cost estimation, empirical modeling and evaluation of complex systems, and intensive data mining for software engineering datasets. He has published papers in prestigious journals including IEEE-TSE, IEEE-SOFTWARE, EMSE and many other leading journals and conferences.



**Akito Monden** received the B.E. degree in 1994 in electrical engineering from Nagoya University, and the M.E. and D.E. degrees in 1996 and 1998 in information science from Nara Institute of Science and Technology. He is Professor in the Graduate School of Natural Science and Technology at Okayama University, Japan. He was honorary research fellow at the University of Auckland, New Zealand (2003–2004). He is a member of the IEEE, ACM, IEICE, IPSJ, and JSSST.



**Kenichi Matsumoto** received the B.E., M.E., and Ph.D. in Information and Computer sciences from Osaka University, Japan, in 1985, 1987, and 1990, respectively. Dr. Matsumoto is currently Professor in the Graduate School of Information Science at Nara Institute of Science and Technology, Japan. His research interests include software measurement and software process. He is an IEICE and IPSJ Fellow, an IEEE senior member, and a member of ACM and JSSST.