A review and comparison of methods for determining the best analogies in analogy-based software effort estimation

Bodin Chinthanet
Faculty of Engineering
Kasetsart University, Thailand
bodin.c@ku.th

Passakorn Phannachitta
Nara Institute of Science and Technology, Japan
phannachitta-p@is.naist.jp

Yasutaka Kamei
Kyushu University, Japan
kamei@ait.kyushu-u.ac.jp

Pattara Leelaprute, Arnon Rungsawang
Faculty of Engineering
Kasetsart University, Thailand
pattara.l@ku.ac.th, arnon@mikelab.net

Naoyasu Ubayashi
Kyushu University, Japan
ubayashi@ait.kyushu-u.ac.jp

Kenichi Matsumoto
Nara Institute of Science and Technology, Japan
matumoto@is.naist.jp

ABSTRACT

Analogy-based effort estimation (ABE) is a commonly used software development effort estimation method. The processes of ABE are based on a reuse of effort values from similar past projects, where the appropriate numbers of past projects (k values) to be reused is one of the long-standing debates in ABE research studies. To date, many approaches to find this k value have been continually proposed. One important reason for this inconclusive debate is that different studies appear to produce different conclusions of the k value to be appropriate. Therefore, in this study, we revisit 8 common approaches to the k value being most appropriate in general situations. With a more robust and comprehensive evaluation methodology using 5 robust error measures subject to the *Wilcoxon rank-sum* statistical test, we found that conflicting results in the previous studies were not mainly due to the use of different methodologies nor different datasets, but the performance of the different approaches are actually varied widely.

CCS Concepts

•Social and professional topics → Pricing and resource allocation;

Keywords

software effort estimation, analogy-based effort estimation, fixed-k, dynamic-k, evaluation criteria

SAC 2016,April 04-08, 2016, Pisa, Italy

 ISBN 978-1-4503-3739-7/16/04...$15.00
DOI: http://dx.doi.org/10.1145/2851613.2851974

1. INTRODUCTION

Accurate software effort estimation is essential for software development projects because it is one of the important factors that will lead the project to its successful completion. The process of software effort estimation is commonly carried out in an early phase of software development where relatively few support information is available for the estimates. Thus, it has become an important and challenging topic in the research communities [5].

In this study, we focus on a commonly used estimation method named analogy-based effort estimation (ABE) [1, 9, 10], a widely accepted method in terms of robustness, accuracy, and reliability [2, 7, 11, 12, 17]. Shepperd and Schofield [16] introduced the principle of ABE: *Projects that are similar with respect to a set of project features will also be similar with respect to the effort.* Derived by this principle, the processes of ABE mainly comprise a search for the most similar past projects, and a reuse of the effort of the past projects. Commonly, similar past projects are identified by calculating and sorting the level of similarity across the dataset. The k most similar past projects will then be the cases whose effort values are reused for the new case estimate. Hence, this can be seen as a form of k-nearest neighbor (kNN) algorithm.

According to our survey, there are many approaches to determine the k value for a local dataset, [1, 2, 9, 10, 7, 11, 12, 17]. Despite being greatly diversified, these approaches are commonly categorized into two groups: (1) statically assign a single k value to the entire local dataset commonly referred to as *fixed-k*, and (2) dynamically learn the k value that fits with the given training/test instance commonly referred to as *dynamic-k*. In this study, we selected 8 common methods, of which 5 are *fixed-k* and the other 3 are *dynamic-k*. The results from many empirical studies [1, 2, 9, 10] endorsed the effectiveness of *dynamic-k* over *fixed-k* methods, but some of which produced conflicting results [10]. This therefore introduces the unstable conclusion, an issue widely recognized in software effort estimation where different studies in the

same area provide greatly diversified conclusions [6, 13].

As suggested by Keung et al. [6], a replicated study with a larger set of data, a more number of more stable performance measures, and a well-controlled sampling method is required to draw a stable conclusion. Therefore, we revisit this question of determining the appropriate number of analogues (k) using 9 standard benchmark datasets. Our performance measures consist only robust error measures suggested by Foss et al. [4] (MAR, $MdAR$, SD, RSD and LSD), and these measure are ensured its statistical significance using the *Wilcoxon rank-sum* test [3]. The *Wilcoxon* test is commonly used in many studies such as in [1, 9, 10]. Our replicated experiments aim at finding a more desirable method to determine the k value for different situations.

The following of this paper is structured to answer this research question:
RQ: What is the more desirable method for determining the best k value currently and at the time of writing?

2. RELATED WORKS

2.1 Analogy-Based Effort Estimation (ABE)

The process of effort estimation based on ABE method begins with a retrieval of the past similar cases. This case retrieval process is to find the level of similarity between the new case and all previous ones. The formulas commonly used to calculate the level of similarity are depicted in Eq.(1) and Eq.(2), where c_i and c_j are two project cases i and j respectively. F is a set of project features of dataset. After retrieval of similar past project cases, an identification of an appropriate number of analogues (k value) is the next imperative process. The important part to find the k value that should produce the most accurate effort estimate for a given dataset. Finally, the averaged value of efforts of the k analogues is the final estimated effort [5].

$$distance(c_i, c_j, F) = \sqrt{\sum_{f \in F} Feature\_dissimilarity(c_{if}, c_{jf})} \quad (1)$$

$$Feature\_dissimilarity(c_{if}, c_{jf}) = \begin{cases} (c_{if} - c_{jf})^2 & \text{if } f \text{ is numeric} \\ 0 & \text{if } f \text{ is categorical } \textit{and } c_{if} = c_{jf} \\ 1 & \text{if } f \text{ is categorical } \textit{and } c_{if} \neq c_{jf} \end{cases} \quad (2)$$

2.2 Determining the Number of Analogues

More recently, research studies have been focusing more on the dynamic selection of the k value. For example, the *Best-k* method proposed by Baker [2] fit a given training dataset to find the appropriate k value in a brute force manner from $k = 1$ to $n - 1$, where n is the total number of project cases in the given training dataset. The common criteria to select if a value k is better than all others is estimation error. In this study we used MAR in our experiments.

Kosti et al. [10] proposed *DD-EbA*, a similar algorithm to *Best-k* with a main different that *DD-EbA* fit only the first nearest neighbour of the new cases rather than the entire training set.

Azzeh and Nassif [1] adopted a hierarchical clustering technique based on k-medoid clustering algorithm [14] to learn the best k value from data distribution. First, the entire dataset is treated as a one single large cluster. Then, this cluster is recursively bisected until the termination criteria is met. Finally, the smallest cluster where the *1NN* of the new case is located become its analogues, the k value is the size of this cluster. The termination criterion [1] is based on cluster compactness (see Eq.3). Compactness is a measure that indicate the degree of similarity within clusters. If the level of Compactness of the parents is lower than that of any subtree, the algorithm is terminated.

$$Compactness = \frac{1}{n} \sum_{i=1}^{k} \sum_{j=1}^{n} \|x_j - v_i\|^2 \quad (3)$$

From the result of the studies [1, 2, 10], conflicting results may hinder practitioners to conclude the best algorithm to select the k value without performing a replication. The followings are the possible reasons: (1) from the study [6], different experimental environments and conditions may produce unstable conclusion. Second, the study of [4] showed that some measures were deprecated and untrustworthy such as $MMRE$, $MMER$, in which many proposed studies still use these measures as evaluation criteria.

3. METHODOLOGY

3.1 Performance Measures

As suggested by Keung et al. [6], we used multiple stable performance measures that proven by Foss et al. [4] to produce a more stable and more robust result. The followings are performance measures used in this study. Let $\hat{x_i}$, and x_i are actual effort and estimated effort of case i:

Absolute Residual (AR):

$$AR_i = |\hat{x_i} - x_i| \quad (4)$$

To summarize the AR, we aggregate it using mean and median. The products are defined as Mean Absolute Residual (MAR) and Median Absolute Residual ($MdAR$):

$$MAR = mean(\text{all } AR) \quad (5) \qquad MdAR = median(\text{all } AR) \quad (6)$$

Alternative to AR, we use standard deviation (SD), Relative standard deviation (RSD) and Logarithmic standard deviation (LSD) as measure of residual error:

$$SD = \sqrt{\frac{\sum_{i=1}^{n} (\hat{x}_i - x_i)^2}{n-1}} \quad (7) \qquad RSD = \sqrt{\frac{\sum_{i=1}^{n} (\frac{\hat{x}_i - x_i}{f_i})^2}{n-1}} \quad (8)$$

$$LSD = \sqrt{\frac{\sum_{i=1}^{n} (e_i - (-\frac{s^2}{2}))^2}{n-1}} \quad (9)$$

where n is a number of case in dataset, f_i is a Function point of case i. For LSD, we describe $e_i = \ln \hat{x_i} - \ln x_i$ and $s^2 = variance(\text{all } e_i)$.

3.2 Datasets

We use the well-known datasets concluded in Table 1. Each dataset contains numeric and categorical features, we can find distance for each type of feature by using Eq.(2).

Note that we normalized all the continuous variables of all the 9 datasets using Eq.(10). This normalization is performed to ensure equal influence across all the computations:

$$normalizedValue_{ij} = \frac{(value_{ij} - min(value_i))}{(max(value_i) - min(value_i))} \quad (10)$$

where i is a feature number and j is a case number.

Table 1: Dataset properties

Dataset	# of project	# of feature	Effort min	Effort max	Effort mean	Effort median
Albrecht	24	8	0.50	105.20	21.88	11.45
Kemerer	15	6	23.20	1107.31	219.25	130.30
Miyazaki94	48	8	5.60	1586.00	87.47	38.10
Finnish	38	5	460.00	26670.00	7678.29	5430.00
Desharnais	77	9	546.00	23940.00	4833.91	3542.00
Cocomo	63	18	5.90	11400.00	683.53	98.00
SDR	12	24	1.00	22.00	5.73	3.50
Nasa93	93	22	8.40	8211.00	624.41	252.00
Maxwell	62	25	583.00	63694.00	8223.21	5189.50

3.3 Evaluation Procedures

In this study, we replicate the algorithm for determining number of analogues (k) including *fixed-k* ($k = 1$ to 5), *Best-k* [2, 9], *DD-EbA* [10], and *BK algorithm* [1]. Hence there are 8 approaches in total being reviewed and evaluated in this study.

For each dataset, first, we applied leave-one-out cross validation test to generate training and test sets. Given n project cases in a dataset, we selected one case as a test set trained by the remaining $n - 1$ cases. Then we applied the 8 approaches to determine the values of k. The result from each method was evaluated by the 5 robust performance measures described earlier in this section. To compare the performance of each approaches, we used the non-parametric statistical named *Wilcoxon rank-sum* test at 95% confidence interval. The *Wilcoxon* test was used for ensuring if the estimation performance were significantly different. The overall performance is assessed using the wins-ties-losses statistic [3]. This statistic is an aggregation of the results across all pairs of method combined in manners of pair-wise comparisons. Algorithm 1 shows the pseudocode to calculate this statistic. Based on this statistic the total sum of *wins*, *losses* and *wins-losses* are the commonly used indicators to compare the overall performance across all approaches being evaluated in software effort estimation research studies such as in [9].

4. PRELIMINARY RESULT

4.1 RQ: What are better methods for determining the appropriate k value at the time of writing?

Table 2 shows the results in terms of *wins*, *losses* and *wins-losses* calculated across the 9 datasets using 5 error measures. From this table, we found that *fixed-k* perform clearly better than *dynamic-k*. Out of the 8 approaches compared in this study, $k = 3$ performed the best in terms of *wins-losses*. Focusing on *dynamic-k*, *Best-k* performed as accurate as the *fixed-k* approaches, while *BK algorithm* was consistently the worst. Therefore these experiments show totally contradictory results from those in the previous study of [1].

```
input  : E → set of error each cases, K → set of measure
output: S → set of N clusters
for all combination case i, j do
    for k in K do
        if WILCOXON(E_ik, E_jk) tells they are same then
            tie_i = tie_i + 1
            tie_j = tie_j + 1
        else
            if E_ik < E_jk then
                win_i = win_i + 1
                loss_j = loss_j + 1
            else
                win_j = win_j + 1
                loss_i = loss_i + 1
            end
        end
    end
end
```

Algorithm 1: wins-ties-losses test

Table 3 shows that there are many cases of Nasa93 dataset that have the same actual effort but the first four k values cannot produce accurate results. As a consequence, the *dynamic-k* approaches such as *DD-EbA* and *BK algorithm*, that build their learning process for the k value based on *1NN*, cannot produce an accurate estimate for these cases.

Table 2: The results in terms of wins, losses, and wins-losses generated across 9 datasets

Method	wins	losses	wins-losses
k = 1	17	3	14
k = 2	16	2	14
k = 3	17	0	17
k = 4	16	0	16
k = 5	13	1	12
Best-k	16	2	14
DD-EbA	14	2	12
BK algorithm	6	105	-99

Table 3: Example of conflicting cases in the Nasa93 dataset

Case number	Actual effort	k = 1	k = 2	k = 3	k = 4
18	60.00	48.00	48.00	40.00	48.00
20	60.00	324.00	192.00	144.00	120.00
21	60.00	60.00	55.00	136.67	131.90
35	60.00	50.00	55.00	136.67	108.80
37	60.00	42.00	78.00	66.00	61.50

There is still no generally best single approach that would performed best in all situations. If we have to suggest one single approach, $k = 3$ would be the single approach of choice.

5. DISCUSSIONS

Based on our experimental results, figures, and tables in this paper, the followings are our findings:

1. All the *fixed-k* methods produced almost the same performance.

2. Even if the *DD-EbA* and *Best-k* approaches are based on a very similar procedure, *DD-EbA* often performs worse. This finding points out that the first nearest neighbor of many project cases may not be sufficiently

similar to the new cases, because the procedure of *DD-EbA* relies the estimation of the new case heavily on its first neighbor. For this situation, data quality assessment and improvement have a strong potential to improve the performance of approaches that rely on the first neighbor.

3. The result in Table 2 indicates that *dynamic-k* performs worse than *fixed-k*. The cause of this result is conflicting data in datasets as shown in Table 3. As suggested by Kocaguneli [8], and Shepperd and Schofield [16], a model built using a better quality data generally provides a better estimation performance. Hence, this finding also pointed out that data quality assessment and improvement has a strong potential to improve the performance of all the *dynamic-k* approaches.

6. CONCLUSIONS

This study revisits the approaches that are being used to determine the best number of analogues (k values), in analogy-based software development effort estimation. Following the suggestions by Shepperd and Kadoda [15], and Menzies et al. [13], we use a large set of data (9 datasets) in our experiment, and use more robust evaluation criteria (5 error measures) [4] subject to *Wilcoxon rank-sum* statistical test. This evaluation procedure is aimed at providing a stable conclusion in this comparison-based study.

Based on our results and findings, we found that *fixed-k* performed generally better than *dynamic-k*. This is contradict to the previous study of [1]. In addition, our analysis of dataset characteristics show that many project cases with similar characteristics were not completed their developments with similar effort values. This finding indicates that data consistency is the essences of *dynamic-k* approaches. Unfortunately, no one to the best of our knowledge has studies the influence of the underlying data quality attribute with the approaches to identified the best number of analogues, reviewed in this study.

Our future work will concentrate on investigating the influence of the two data quality attributes, data homogeneity and data consistency. We strongly believe that the improvement of these two attributes will significantly *dynamic-k* approaches. And since *dynamic-k* are more theoretically driven and being more practical, we believe that research studies in this direction will allow us to address the single best improved method to determine the best number of analogues, an important factor associate with the accuracy of analogy-based software effort estimation.

7. ACKNOWLEDGMENTS

The first author would like to thank both Kasetsart University and NAIST-Japan for giving him a good opportunity to be included in KU-NAIST internship 2015 program.

8. REFERENCES

[1] M. Azzeh and A. B. Nassif. Analogy-based effort estimation: a new method to discover set of analogies from dataset characteristics. *Software, IET*, 9(2):39–50, 2015.

[2] D. R. Baker. *A Hybrid Approach to Expert and Model Based Effort Estimation*. ProQuest, 2007.

[3] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, Dec. 2006.

[4] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit. A simulation study of the model evaluation criterion mmre. *IEEE Trans. Softw. Eng.*, 29(11):985–995, Nov. 2003.

[5] J. Keung. Software development cost estimation using analogy: A review. In *Proceedings of the 2009 Australian Software Engineering Conference*, 2009.

[6] J. Keung, E. Kocaguneli, and T. Menzies. Finding conclusion stability for selecting the best effort predictor in software effort estimation. *Automated Software Eng.*, 20(4):543–567, Dec. 2013.

[7] C. Kirsopp, E. Mendes, R. Premraj, and M. Shepperd. 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*, 2003.

[8] E. Kocaguneli, G. Gay, T. Menzies, Y. Yang, and J. W. Keung. When to use data from other projects for effort estimation. In *Proceedings of the IEEE/ACM*, ASE '10, 2010.

[9] E. Kocaguneli, T. Menzies, A. Bener, and J. W. Keung. Exploiting the essential assumptions of analogy-based effort estimation. *IEEE Trans. Softw. Eng.*, 38(2):425–438, Mar. 2012.

[10] M. V. Kosti, N. Mittas, and L. Angelis. Dd-eba: An algorithm for determining the number of neighbors in cost estimation by analogy using distance distributions. In *Proceedings of the 3rd Artificial Intelligence Techniques in Software Engineering Workshop*, 2010.

[11] Y. F. Li, M. Xie, and T. N. Goh. A study of the non-linear adjustment for analogy based software cost estimation. *Empirical Softw. Eng.*, 14(6):603–643, Dec. 2009.

[12] E. Mendes, N. Mosley, and S. Counsell. 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*, 2003.

[13] T. Menzies, O. Jalali, J. Hihn, D. Baker, and K. Lum. Stable rankings for different effort models. *Automated Software Eng.*, 17(4):409–437, Dec. 2010.

[14] H. S. Park and C. H. Jun. A simple and fast algorithm for k-medoids clustering. *Expert Syst. Appl.*, 36(2):3336–3341, Mar. 2009.

[15] M. Shepperd and G. Kadoda. Comparing software prediction techniques using simulation. *IEEE Trans. Softw. Eng.*, 27(11):1014–1022, Nov. 2001.

[16] M. Shepperd and C. Schofield. Estimating software project effort using analogies. *IEEE Trans. Softw. Eng.*, 23(11):736–743, Nov. 1997.

[17] F. Walkerden and R. Jeffery. An empirical study of analogy-based software effort estimation. *Empirical Softw. Eng.*, 4(2):135–158, June 1999.