Sampling Methods in Software Effort Estimation: An Investigation on Bias-Variance Trade-Off

Ekrem Kocaguneli, *Student Member, IEEE* Tim Menzies, *Member, IEEE* Martin Shepperd, *Member, IEEE*

Abstract—
Background: More than half the software effort estimation studies focus on model comparisons, for which sampling methods are fundamental. Studies adopt different sampling methods such as leave-one-out (LOO), 10-Way, 3-Way cross-validation. However, none of the studies that we are aware of empirically justifies its sampling method. ***Aim:*** According to theory more and smaller test sets, increase the variance and decrease the bias. However, there is no empirical investigation of this theoretical concept in software effort estimation. In this paper, we empirically investigate whether theoretical assumptions hold for software effort datasets. ***Method:*** We evaluated 90 different algorithms on 20 different effort datasets. For each algorithm, we calculated the bias and variance values under LOO, 3Way and 10Way to check if they behave as predicted by the theory. ***Results:*** This extensive study showed that the theory does not hold for effort estimation datasets. We observed that the majority of the methods have statistically same bias and variance values under different sampling methods. ***Conclusion:*** Since effort datasets used here are indifferent to sampling methods, we conclude that the bias-variance trade-off is not the main concern for selecting a sampling method. When opting for a particular sampling method, we recommend consideration of run-times and reproducibility of the results.

Index Terms—Software Cost Estimation, Experimentation, Bias, Variance

1 INTRODUCTION

Sampling method is an important topic for software effort estimation (from now on SEE) studies and an empirical study to compare the pros and cons of different sampling methods in SEE is urgent.

The biggest research topic in SEE since 1980s is the introduction and comparison of new methods [**?**]. In their comprehensive systematic review Jorgensen and Shepperd report 61% of selected SEE papers deal with that topic [**?**]. This group of papers use *historical data*, i.e. and not a single one of them employs a data collection methodology.

Ideally a learned theory should be applied to new scenarios to observe if the predicted effect occurs in practice. Only generating theories from historical data entails an internal validity threat, which we would like to call *"fixed-scenario-problem"*. *fixed-scenario-problem* is the lack of new scenarios in evaluation and it threats the evaluation experiments like the ones reported in [**?**]. Therefore, studies without a new scenario for the learned model are limited within their experimental settings.

On the other hand it is impractical to expect every study to collect new data. The mitigation to *fixed-scenario-problem* is possible by simulating the application of a method to a new situation. A sampling method (from now on SM) forms the basis of such a simulation [**?**], [**?**].

There is a wide palette of available SMs used in the literature: Leave-one-out (LOO), 3Way and 10Way are some examples to the most commonly used ones [**?**], [**?**], [**?**], [**?**]. Similar to choosing colors from a palette, the choice of different SMs results in a different picture. For example, theoretically LOO results in high-variance and low-bias in the results, whereas 3Way generates just the opposite (low-variance, high-bias) and 10Way in between the two methods [**?**], [**?**]. The change from one method to the other is known as bias and variance (from now on $B\&V$) trade-off. Employment the right SM and correct interpretation of $B\&V$ trade-off is crucial to the validity of a particular study.

To our surprise, in SEE domain there is no study employing a rigorous experimentation to observe the effects of different SMs. Kitchenham et al. have already identified and raised the issue of SM selection [**?**]; however, their mentioning is more of a pointer to future work rather than an investigation. Hence, this paper is a natural follow-up of previous SEE studies. Furthermore, it is the first of its kind to rigorously investigate the $B\&V$ trade-off inherent in different SMs and it concerns more than half the SEE field. In this paper, we present $B\&V$ trade-off results of 3 different SMs: LOO, 3Way and 10Way cross-validation. Our experimentation includes 90 methods applied on 20 datasets.

The results show that $B\&V$ behavior of SMs are different than the predicted: For the majority of the algorithms, $B\&V$ values of LOO, 3Way and 10Way are statistically the same. However, we have seen orders of magnitude differences in terms of run times, see Figure 1 for exact values. The values of Figure 1 belong to experiments coded in MATLAB and run on a 64-bit dual-core machine. Given these findings, we recommend focusing on experimental concerns to choose an SM. If the main concern is the exact reproduction of the current work, then LOO is to be used. Otherwise, if the

- *Ekrem Kocaguneli and Tim Menzies are with the Lane Department of Computer Science and Electrical Engineering, West Virginia University. E-mail: ekocagun@mix.wvu.edu, tim@menzies.us*
- *Martin Shepperd is with the School of Information Systems Computing & Maths, Brunel University. E-mail: martin.shepperd@brunel.ac.uk*

This research is funded in part by NSF,CISE, project #0810879

SM	Run Time
LOO	More than $86,400$
3Way	$14,280$
10Way	$28,860$

Fig. 1. The run times in seconds for different SMs. Note that LOO is orders of magnitude slower than 3Way and 10Way. These run times are for all the datasets included. Execution of LOO was terminated after 24 hours.

lower run times are the main concern, then we recommend 3Way or 10Way. 3Way generates lower run times than 10Way, but 10Way is more commonly used in SEE (see Figure 3 for mapping of studies to SMs). Therefore, unless 10Way is prohibitively slower (days of run times) than 3Way, we recommend 10Way over 3Way.

1.1 Contributions

The contributions of this research are:

- The first systematic investigation of *B&V* trade-off in SEE domain
- An extensive experimentation of 20 datasets and 90 algorithms
- Showing that *B&V* trade-off is not the main concern for SEE
- Recommendations based on experimental concerns:
  - For lower run-times the order of preference is: 1) 10Way, 2) 3Way, 3) LOO.
  - For reproducibility prefer LOO

2 Terminology

A typical SEE dataset consists of a matrix X and a vector Y. The input variables (a.k.a. features) are stored in X, where each row corresponds to an observation and each column corresponds to a particular variable. Similarly, the dependent variable is stored in a vector Y, where for each observation in X there exists a response value.

Now assume that a prediction model represented by $\hat{f}(x)$ has been learned from a training dataset and the actual values in the training set were generated by an unknown function $f(x)$. So as to measure the errors between the actual values in Y and the predictions given by $\hat{f}(x)$, we can make use of an error function represented by $L(Y, \hat{f}(x))$. Some examples of error functions are squared loss (Equation 1) and absolute loss (Equation 2).

$$L(Y, \hat{f}(x)) = \left(Y - \hat{f}(x)\right)^2 \tag{1}$$

$$L(Y, \hat{f}(x)) = |Y - \hat{f}(x)| \tag{2}$$

Given the assumptions that the underlying model is $Y = f(X) + \epsilon$ where $E(\epsilon) = 0$ and $Var(\epsilon) = \sigma_\epsilon^2$, then we can come up with a derivation of the squared-error loss for $\hat{f}(X)$ [**?**]. The error for a point $X = x_0$ is:

$$\begin{aligned} Error(x_0) &= E\left[\left(Y - \hat{f}(x_0)\right)^2 | X = x_0\right] \\ &= \sigma_\epsilon^2 + \left(E[\hat{f}(x_0) - f(x_0)]\right)^2 \\ &\quad + E\left[\hat{f}(x_0) - E[\hat{f}(x_0)]\right] \\ &= \sigma_\epsilon^2 + Bias^2(\hat{f}(x_0)) + Var(\hat{f}(x_0)) \\ &= \underbrace{IrreducableError}_{1^{st}Term} + \underbrace{Bias^2}_{2^{nd}Term} \\ &\quad + \underbrace{Variance}_{3^{rd}Term} \end{aligned}$$

In the above derivation, the explanations of the 1^{st}, 2^{nd} and 3^{rd} terms are as follows:

- The $1^{st}Term$ is the so called *"irreducable error"*, i.e. the variance of the actual model around its true mean. This variance is inevitable regardless of how well we model $f(x_0)$, only exception to that is when the actual variance is zero (when $\sigma_\epsilon^2 = 0$).
- The $2^{nd}Term$ is the square of the bias, which is the measure of how different the model estimates are fromt the *true* mean of the underlying model.
- The $3^{rd}Term$ is the variance of the estimated model. It is the expectation of the squared deviation of the estimated model from its own mean.

Furthermore, the above derivation is for an individual instance. The $B\&V$ values associated with an algorithm $\hat{f}(X)$ is the mean of all individual values.

An important question associated with SMs is how $B\&V$ relate to different choices of the training size (*K*). In this study we consider three different SMs: LOO, 3Way and 10Way. These SMs are selected due to their frequent use in SEE (see Figure 4). The brief descriptions of SMs applied on a dataset of size N are:

- LOO
  - Take one instance at a time as the test set
  - Build the learner on the remaining $N - 1$ instances (training set)
  - Use the model to estimate for the test set
- 3Way
  - Randomize order of rows in data
  - Divide dataset into 3 subsets of size close or equal to $N/3$
  - Use each subset as the test set and the remaining subsets as the training set
  - Repeat above procedure 10 times
- 10Way
  - Randomize order of rows in data
  - Divide dataset into 10 subsets of size close or equal to $N/10$
  - Use each subset as the test set and the remaining subsets as the training set

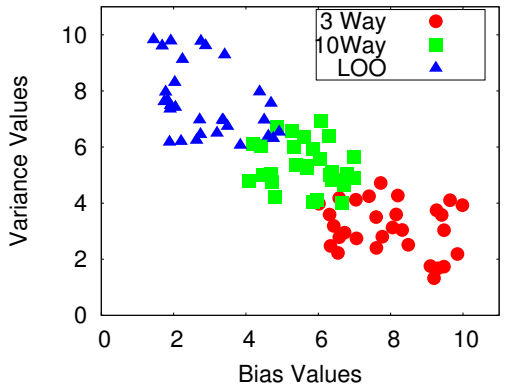

Fig. 2. A simple simulation for the *"expected"* case of $B\&V$ relation to testing strategies.

– Repeat above procedure 10 times

"Theoretically" when training size (K) is equal to the dataset size (K=N), we expect SM to be approximately unbiased and to have high variance, because N training sets are so similar to one another and N models are used for estimation. On the other hand, for small values of K, say K=$N/3$ as in 3Way, we expect low variance and high bias [**?**]. For an SM that has training sizes between LOO and 3Way, say K=$N/10$ as in 10Way, expected $B\&V$ values are between those of LOO and 3Way. Naively put, the *"expected"* relationship is:

- LOO : High variance, low bias
- 3Way : Low variance, high bias
- 10Way : Values between LOO and 3Way

In an ideal case, when we plot $B\&V$ values of each individual test instance on x and y axes respectively, we expect 3 clusters:

- Upper Left : Low bias, high variance; i.e. LOO results.
- Lower right : High bias, low variance; i.e. 3Way results.
- Center : In-between $B\&V$; i.e. 10Way results.

A very *simple* but *ideal* case would look like Figure 2. Figure 2 is only demonstrative and $B\&V$ values are randomly generated.

3 Related Work

3.1 Bias-Variance Trade-Off

The SMs and the $B\&V$ trade-off are critical to assess different learners [**?**], [**?**], [**?**], [**?**]. The issue has been both theoretically [**?**], [**?**], [**?**] as well as practically investigated [**?**], [**?**], [**?**], [**?**]. It is our understanding that the theoretically *expected* and *actual* behavior of different SMs may or may not overlap. Depending on dataset size and type, actual $B\&V$ values may be different than *expected*.

The size and type of effort estimation datasets are quite unique: noisy and limited in size [**?**], [**?**]. Kitchenham and Mendes have discussed the effects of SMs on $B\&V$ in the context of cross-within-company data [**?**] and suggested that LOO biases positively towards within-company data. However, their discussion is based on their extensive expert knowledge of the area and does not include an experimentation. Aside from Kitchenham et al.'s study [**?**], there is no other SEE study investigating the $B\&V$ trade-off associated with SMs.

Dataset	Used by
telecom	[**?**], [**?**]
kemerer	[**?**], [**?**], [**?**]
cocomo81o	[**?**], [**?**], [**?**] ,
desharnaisL1	[**?**] ,
cocomo81s	[**?**], [**?**], [**?**] ,
desharnaisL3	[**?**] ,
albrecht	[**?**], [**?**], [**?**], [**?**], [**?**], [**?**]
cocomo81e	[**?**], [**?**], [**?**]
nasa93_center_5	[**?**], [**?**], [**?**]
desharnaisL2	[**?**]
desharnais	[**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**]
maxwell	[**?**], [**?**]
sdr	[**?**], [**?**]
nasa93_center_1	[**?**], [**?**], [**?**]
miyazaki94	[**?**]
nasa93_center_2	[**?**], [**?**], [**?**]
finnish	[**?**], [**?**]
cocomo81	[**?**], [**?**], [**?**], [**?**],
nasa93	[**?**], [**?**], [**?**]
china	this study

Fig. 3. A sample of effort estimation papers that use the data sets explored in this paper.

Method	Used by
LOO	[**?**], [**?**], [**?**] [**?**], [**?**], [**?**] [**?**], [**?**]
Others (ad-hoc, 6-Way etc.)	[**?**], [**?**], [**?**] [**?**], [**?**], [**?**]
10-Way	[**?**], [**?**], [**?**] [**?**]
3-Way	[**?**]

Fig. 4. Distribution of the studies in Figure 3 w.r.t. their SM. Majority of the studies use LOO. LOO is followed by ad-hoc methods, 10-Way then 3-Way.

In Figure 3 we list the studies that use one or more of the datasets used in our work, then in Figure 4 we grouped all these studies according to their adopted SM. Note that we did not include the studies of Figure 3 into Figure 4 if no explicit statement of the adopted SM was made. Figure 4 shows that LOO and 10Way are the most popular SMs, followed by other ad-hoc methods, then by 10Way and then by 3-Way. A wide range of different SMs are adopted in SEE and most of the time without a justification. Therefore, an empirical investigation of SMs is critical and urgent; hence, the rest of this paper.

3.2 Effort Estimation Methods

Effort estimation is the activity of predicting the amount of effort required to complete a software development project [**?**]. Estimation activities are carried out through:

- algorithmic methods
- non-algorithmic methods

Algorithmic methods learn a model from historical data and pass new projects through that model to generate their estimates. The number of proposed algorithmic methods and associated variants easily exceed tens of thousands. Figure 3 of [**?**] shows that for analogy-based effort estimation (which is just one branch of algorithmic methods), likely combinations are more than 6000. Some other examples to algorithmic

methods are: various kinds of regression (simple, partial least square, stepwise, regression trees), neural networks and instance-based algorithms, just to name a few. In Appendix A, we provide the algorithmic methods used in this study.

Non-algorithmic methods utilize the best knowledge of experienced human experts. Such non-algorithmic methods, a.k.a. expert-based estimation, is defined to be a human intensive approach that is most commonly adopted in practice [**?**]. In expert-based variants, estimates are produced by domain experts based on their very own personal experience. On one hand, these methods are flexible and intuitive as they can be applied in a variety of circumstances where other estimating techniques do not work. For example, when there is no historical data or the requirements of a project are unavailable at the initial stages, a rough estimate in a very short period of time can be provided by expert estimates. On the other hand -regardless of the efforts to establish guidelines for expert-based methods [**?**]- there are still many ad-hoc methods used in practice. Shepperd et al. [**?**] do not consider expert based estimation as an empirical method, since the means of deriving an estimate are not explicit and therefore not repeatable, nor easily transferable to other staff. In addition, knowledge relevancy is also a problem, as an expert may not be able to justify estimates for a new application domain. Lastly, from an experimental point of view SMs do not make sense for expert estimates, because expert estimates are based on the expert's personal experience rather than different divisions of train/test sets. Hence, the rest of this paper excludes non-algorithmic methods from the discussion of $B\&V$.

4 METHODOLOGY

4.1 Algorithms: Pre-Processors & Learners

This study uses *10 different pre-processors* $\times$ *9 learners = 90 algorithms*. The selection is based on two criteria:

- Learners and pre-processors must come from SEE literature; e.g. [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**].
- Learners must make different assumptions about the data.

This second criteria is based on data-mining theory that different learners are built on different assumptions, hence they have different biases [**?**], [**?**], [**?**], [**?**].

We hence used 10 pre-processors:

- Three *simple preprocessors*: **none, norm, and log**;
- One *feature synthesis* method: **PCA**;
- Two *feature selection* methods: **SFS** and **SWreg**;
- Four *discretization* methods: Based on equal frequency/width.

and 9 learners:

- Two *iterative dichotomizers*: **CART(yes), CART(no)**;
- A *neural net*: **NNet**;
- Four *regression methods*: **LReg, PCR, PLSR, SWReg**.
- Two *instance-based* learners: **ABE0-1NN, ABE0-5NN**;

Note that "ABE" is short for analogy-based effort estimation. ABE0-kNN is a standard analogy-based estimator with execution steps of:

- *Normalization* of data to zero-one interval;
- A *Euclidean* distance measure;
- Estimates generated using the *k nearest neighbors*.

For detailed descriptions of all these learners, see Appendix A.

4.2 Experiments

Get estimates: Let A_i ($i \in \{1, 2, ..., 90\}$) be one of the 90 algorithms and let D_j ($j \in \{1, 2, ..., 20\}$) be one of the 20 datasets. Also let SM_k ($k \in \{1, 2, 3\}$) be one of the 3 SMs. In this step every A_i is run on every D_j subject to every SM_k. In other words every $A_i \times D_j \times SM_k$ combination is exhausted, and actuals as well as related predictions are stored to be used for $B\&V$ calculations. As it is impractical for many practitioners to wait the run of an estimator more than a day, the LOO experiments including *"china"* dataset (with 499 instances) were stopped after the run time exceeded one day. Hence, we did not include $B\&V$ values of *china* dataset in §5, but we make us of its run-times.

Calculate $B\&V$ values: The actuals and predictions coming from $A_i \times D_j \times SM_k$ runs are used to calculate the $B\&V$. At the end of this step, we have 1 array of *individual* $B\&V$ values for every $A_i \times D_j \times SM_k$. Another interpretation is that for every algorithm-dataset combination ($A_i \times D_j$) we have 3 arrays of $B\&V$ values (1 for each SM_i). The *overall* $B\&V$ values associated with every $A_i \times D_j \times SM_k$ are the mean of individual values.

Statistical Check on $B\&V$ values: In this step we check if the 3 arrays of $B\&V$ values for every $A_i \times D_j$ combination are statistically different from one another (checks are based on Mann-Whitney at 95% confidence interval). This way we can see if the run of an algorithm on a single dataset subject to different SMs generate significantly different $B\&V$ values. Since we have 3 different SMs, for every $A_i \times D_j$ there are 3 different tuples to look at: LOO vs. 3Way, LOO vs. 10Way, 3Way vs. 10Way. When we are done with all the $A_i \times D_j$ combinations, we also see what percent of the 90 algorithms resulted in significantly different $B\&V$ values.

Observe Run Times: The total execution time of the experimentation is associated with particular implementation method, i.e. different implementations of the same algorithm will have different run times. Therefore, we used standard MATLAB functions in this study: All methods except ABE0-1NN and ABE0-5NN, and all pre-processors except discretizers are found in MATLAB libraries.

The run times are also greatly affected by particular SMs. Each SM dictates a different number of times a learner is trained. The training-time of a learner is much greater than the testing-time. Once a learner is trained, the prediction for a particular test instance is instantaneous. Below are the number training times required for each SM:

- **LOO:** *N trains* where N is the dataset size.
- **3Way:** *10 repeats x 3 bins = 30 trains*
- **10Way :** *10 repeats x 10 bins = 100 trains*

From above number of training times, we expect 3Way to be the fastest SM. The difference between LOO and 10Way is supposed to be greatly influenced by the dataset size: Until 100 instances, we expect LOO and 10Way to produce comparable run times, whereas after 100 instances we expect LOO to be much slower.

Dataset	Features	Size	Description	Historical Effort Data					
				Units	Min	Median	Mean	Max	Skewness
cocomo81	17	63	NASA projects	months	6	98	683	11400	4.4
cocomo81e	17	28	Cocomo81 embedded projects	months	9	354	1153	11400	3.4
cocomo81o	17	24	Cocomo81 organic projects	months	6	46	60	240	1.7
cocomo81s	17	11	Cocomo81 semi-detached projects	months	5.9	156	849.65	6400	2.64
nasa93	17	93	NASA projects	months	8	252	624	8211	4.2
nasa93_center_1	17	12	Nasa93 projects from center 1	months	24	66	139.92	360	0.86
nasa93_center_2	17	37	Nasa93 projects from center 2	months	8	82	223	1350	2.4
nasa93_center_5	17	40	Nasa93 projects from center 5	months	72	571	1011	8211	3.4
desharnais	12	81	Canadian software projects	hours	546	3647	5046	23940	2.0
desharnaisL1	11	46	Projects in Desharnais that are developed with Language1	hours	805	4035.5	5738.9	23940	2.09
desharnaisL2	11	25	Projects in Desharnais that are developed with Language2	hours	1155	3472	5116.7	14973	1.16
desharnaisL3	11	10	Projects in Desharnais that are developed with Language3	hours	546	1123.5	1684.5	5880	1.86
sdr	22	24	Turkish software projects	months	2	12	32	342	3.9
albrecht	7	24	Projects from IBM	months	1	12	22	105	2.2
finnish	8	38	Software projects developed in Finland	hours	460	5430	7678.3	26670	0.95
kemerer	7	15	Large business applications	months	23.2	130.3	219.24	1107.3	2.76
maxwell	27	62	Projects from commercial banks in Finland	hours	583	5189.5	8223.2	63694	3.26
miyazaki94	8	48	Japanese software projects developed in COBOL	months	5.6	38.1	87.47	1586	6.06
telecom	3	18	Maintenance projects for telecom companies	months	23.54	222.53	284.33	1115.5	1.78
china	18	499	Projects from Chines software companies	hours	26	1829	3921	54620	3.92
		Total: 1198							

Fig. 5. The 1198 projects used in this study come from 20 data sets. Indentation in column one denotes that indented dataset is a subset of another one.

4.3 Datasets

There is at least one study in SEE using one or more of the 20 datasets used in our study (see Figure 3). Therefore, the results presented here are based on a large corpus and concern a number of previously published SEE studies. The description of 20 datasets used in this study are provided in Figure 5. These datasets are available at http://promisedata.org/data.

As described in Figure 5, the datasets were collected in different parts of the world:

- The desharnais dataset includes Canadian software projects,
- cocomo81 and nasa93 include projects developed in the United States,
- sdr, contains projects of various software companies in Turkey [**?**].

Note that three of these data sets (nasa93_center_1, nasa93_center_2, nasa93_center_5) come from different development centers around the United States. Another three of these data sets (cocomo81e, cocomo81o, cocomo81s) represent different kinds of projects (embedded, organic and semi-detached respectively) developed by different team sizes and under different constraints [**?**].

Note also in Figure 5, the skewness of the effort values (up to 6.06): The datasets are extremely heterogeneous with as much as 60-fold variation. There is also some divergence in the features used to describe the datasets:

- While data sets have some effort values in common (measured in terms of man-months or man-hours), no other feature is shared by all data sets.
- The cocomo* and nasa* data sets use the features defined by Boehm [**?**]; e.g. analyst capability, required software reliability, memory constraints, and use of software tools.
- The other data sets use a wide variety of features including, number of entities in the data model, number of basic logical transactions, query count and number of distinct business units serviced.

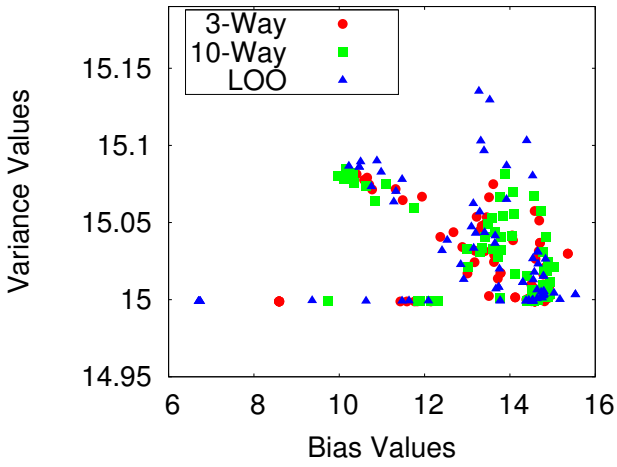

Fig. 6. $B\&V$ values for cocomo81.

5 Results

After calculating the $B\&V$ values for 90 algorithms on all the datasets, we were unable to observe the behavior of Figure 2, i.e. we did not observe three clusters at predicted $B\&V$ zones. On the contrary, we observed that $B\&V$ values associated with different SMs were very close to one another.

For example, see in Figure 6 the mean $B\&V$ values of 90 algorithms for cocomo81 dataset. Note that different SMs are represented with different symbols and for every SM there are 90 symbol occurrences corresponding to 90 algorithms. The $B\&V$ values associated with each SM overlap, instead of forming separate clusters. Also, the *expected* relative low and high $B\&V$ values of SMs (see Figure 2 for expected low and high) were not visible too. Unlike the expected behavior, the actual $B\&V$ values were both high, regardless of the utilized SM.

All the values reported in Figure 6 are logged. Also note that the axes are not scaled to an interval (say 1 to 20), because

dataset		bias		variance	
		3Way	10Way	3Way	10Way
cocomo81	LOO	43	82	57	80
	3Way	-	21	-	40
cocomo81o	LOO	91	100	76	93
	3Way	-	90	-	63
cocomo81e	LOO	68	89	54	78
	3Way	-	36	-	19
cocomo81s	LOO	62	87	56	74
	3Way	-	32	-	34
nasa93	LOO	81	90	62	76
	3Way	-	59	-	60
nasa93_center_1	LOO	94	94	47	84
	3Way	-	81	-	47
nasa93_center_2	LOO	84	96	77	91
	3Way	-	58	-	42
nasa93_center_5	LOO	87	97	70	88
	3Way	-	71	-	41
desharnais	LOO	100	100	91	93
	3Way	-	100	-	81
desharnaisL1	LOO	100	100	91	92
	3Way	-	98	-	86
desharnaisL2	LOO	99	100	91	93
	3Way	-	94	-	69
desharnaisL3	LOO	94	100	60	100
	3Way	-	86	-	43
sdr	LOO	52	64	29	62
	3Way	-	20	-	17
albrecht	LOO	99	100	79	93
	3Way	-	78	-	50
finnish	LOO	100	100	91	92
	3Way	-	100	-	84
kemerer	LOO	92	100	78	86
	3Way	-	82	-	58
maxwell	LOO	94	100	81	89
	3Way	-	82	-	64
miyazaki94	LOO	77	93	52	78
	3Way	-	50	-	36
telecom	LOO	100	100	91	96
	3Way	-	100	-	70

Fig. 8. Percentage of algorithms for which $B\&V$ values coming from different SMs are the same. Note the very high percentage values, meaning that for the majority of the algorithms different SMs generate statistically the same values.

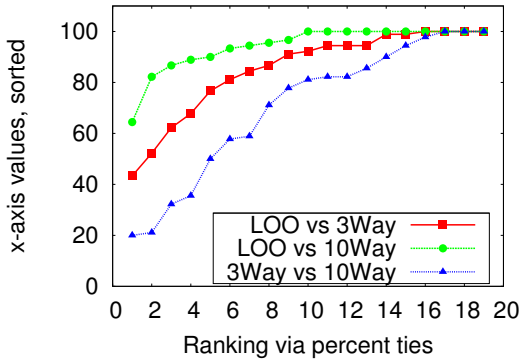

Fig. 9. Sorted bias values of LOO, 3Way and 10Way. Actual values are given in Figure 8.

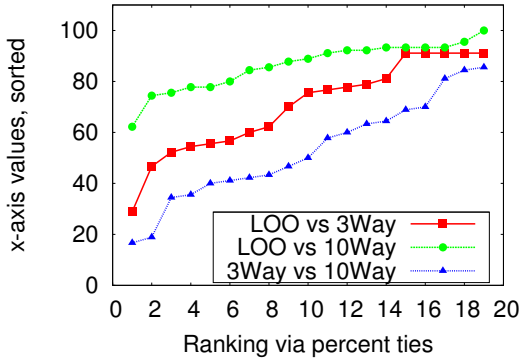

Fig. 10. Sorted variance values of LOO, 3Way and 10Way. Actual values are given in Figure 8.

the differences are so small that scaling the axes makes it impossible to observe the behavior of $B\&V$ values.

We have conducted these experiments on all the datasets and generated Figure 6 for every dataset. However, the results are the same:

- 1) No *expected* behavior of LOO, 3-Way and 10Way;
- 2) No distinct clusters, i.e. overlapping $B\&V$ values of LOO, 3-Way and 10Way.

As plots do not provide additional information and as repeating Figure 6 for every dataset is not possible due to space constraints, we summarized these $B\&V$ values in terms of quartile charts in Figure 7. Figure 7 shows every dataset in a separate row, which is then divided into 3 sub-rows. Sub-rows correspond to 3 different SMs and they show the related $B\&V$ quartile charts separately. In every quartile chart, the median (represented with a dot), 25^{th} quartile (the left horizontal line-end) and the 75^{th} quartile (the right horizontal line-end) are shown. Note in Figure 7, how median as well as the quartiles of different SMs occur on top of one another. In other words, Figure 7 proves our findings from Figure 6 in a much larger scale.

Another way to investigate the $B\&V$ values associated with different SMs is to check their statistical significance. Figure 8 shows what percent of 90 algorithms had statistically the same bias or variance values coming from tuples of SMs. Every cell of Figure 8 reports the percentage associated with a comparison between tuples of SMs (LOO vs. 3Way, LOO vs. 10Way, 3Way vs. 10Way). Note the extremely high percentage values in Figure 8, meaning that for a very high percent of the algorithms, the difference in $B\&V$ values coming from different SMs are statistically insignificant.

Figure 9 and Figure 10 provide another perspective to Figure 8: Sorted percentages of every SM tuple in Figure 8. See in both figures that the percentage values of 3Way vs. 10Way are lowest, whereas the percentages associated with LOO vs. 3Way and LOO vs. 10Way are much higher. $B\&V$ values of LOO are much closer to 3Way and 10Way. However, the $B\&V$ values of 3Way and 10Way more separate from one another.

In Figure 5 we see run times for two different settings.

- All datasets included:A total of 1198 instances, where the largest dataset (*china*) has 499 instances.
- *china* dataset excluded: A total of 699 instances, where the largest dataset has 93 instances.

Our expectation that LOO would be much slower than 3Way and 10Way for large datasets turned out to be true. When 3Way and 10Way finished execution on 1198 instances within

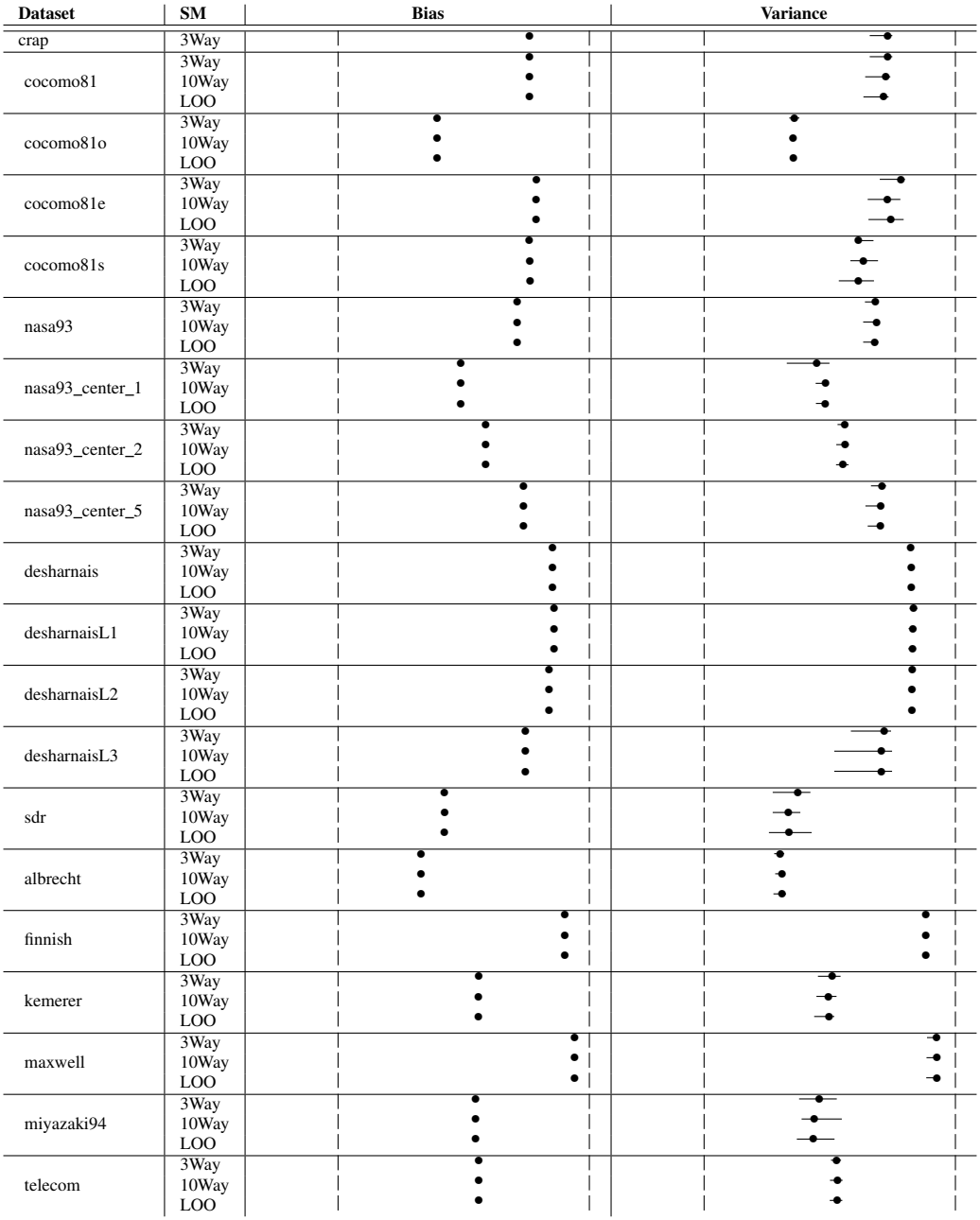

Fig. 7. $B\&V$ values in quartiles for all datasets.

the course of hours, LOO was still not finished with *china* dataset at the end of an entire day. Another expected result that LOO and 10Way would have comparable run times for datasets of size up to 100 also turned out to be correct. Note the execution times of LOO and 10Way for 699 instances are very close. Finally, since 3Way requires the least amount of learner training, its execution is the fastest in both cases.

6 Threats to Validity

One obvious threat to the validity of our results is the implementation of the algorithms. Although we used standard functions from MATLAB libraries, there is still considerable code into which standard functions were embedded. Therefore, run-times will be different in other implementations. However, since all SMs are run on the same code-base, the relative position of SMs in terms of run-times would remain the same.

Another validity threat concerning the run-times is the particular machine on which the experiments are run. Similar to implementation, different machines will yield different run-

SM	499/1198 instances	93/699 instances
LOO	More than $86,400$	$4,980$
3Way	$14,280$	$3,720$
10Way	$28,860$	$5,280$

Fig. 11. The run times in seconds. Column name convention is: *largest dataset size*/*total number of instances*. Execution of LOO was terminated after 24 hours.

times, but the relative position of SMs will remain the same.

The choice of SMs in this work depends on the literature using one or more of datasets used here. On the other hand, different choices of SMs may also have an effect on the results. We fully support that claim and leave it as a future work. However, we also remind that this study based on 90 algorithms and 20 datasets is much more extensive than most of the SEE papers.

7 Conclusions

This study is a natural result of the prior work on SEE and it relates to more than half of the field. To the best of our knowledge, it is the first empirical investigation on $B\&V$ trade-off inherent in different SMs.

Our experimentation investigates a very large space of 90 algorithms and 20 datasets. The results present the surprising finding that $B\&V$ values in SEE domain behave quite different than the expected: Different SMs are statistically the same. However, there is no similarity between SMs in terms of their run times. On the contrary, there are orders of magnitude differences.

We finish this study with recommendations based on our empirical findings. For reproducibility purposes the suitable SM turns out to be LOO. For lower run times the SMs to be used -in the order of preference- are 10Way and 3Way.

8 Future Work

The most likely future direction to this work is the reproduction of it with new datasets (and possibly with new algorithms) to see if the $B\&V$ behavior persists.

Another future direction is to investigate further, why theoretical and actual behavior of $B\&V$ differs from one another in SEE datasets.

Finally, repeating the $B\&V$ analysis reported here with other SMs in SEE (including the ad-hoc methods) and reaching a consensus to determine which SMs should be used under which conditions is the ultimate goal of this initial analysis.

Appendix

A. Methods

This study uses 90 algorithms, which are product of 10 pre-processors combined with 9 learners. The details of the learners as well as the pre-processors are provided below.

A.1. Ten Pre-processors

In this study, we investigate:

- Three *simple preprocessors*: **none, norm, and log**;
- One *feature synthesis* methods called **PCA**;
- Two *feature selection* methods: **SFS** (sequential forward selection) and **SWreg**;
- Four *discretization* methods: divided on equal frequency/width.

None is the simplest preprocessor- all values are unchanged.

With the **norm** preprocessor, numeric values are normalized to a 0-1 interval using Equation 3. Normalization means that no variable has a greater influence than any other.

$$normalizedValue = \frac{(actualValue - min(allValues))}{(max(allValues) - min(allValues))} \quad (3)$$

With the **log** preprocessor, all numerics are replaced with their natural logarithm value. This **log**ging procedure minimizes the effects of the occasional very large numeric values.

Principal component analysis [**?**], or **PCA**, is a *feature synthesis* preprocessor that converts a number of possibly correlated variables into a smaller number of uncorrelated variables called components. The first component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible.

Some of the preprocessors aim at finding a subset of all features according to certain criteria such as **SFS** (sequential forward selection) and **SWR** (stepwise regression). **SFS** adds features into an initially empty set until no improvement is possible with the addition of another feature. Whenever the selected feature set is enlarged, some oracle is called to assess the value of that set of features. In this study, we used the MATLAB, *objective* function (which reports the the mean-squared-error of a simple linear regression on the training set). One caution to be made here is that exhaustive search algorithms over all features can be very time consuming (2^n combinations in an *n*-feature dataset), therefore SFS works only in forward direction (no backtracking).

SWR adds and removes features from a multi-linear model. Addition and removal is controlled by the p-value in an F-Statistic. At each step, the F-statistics for two models (models with/out one feature) are calculated.

Discretizers are pre-processors that maps every numeric value in a column of data into a small number of discrete values:

- **width3bin:** This procedure clumps the data features into 3 bins, depending on equal width of all bins see Equation 4.

$$binWidth = ceiling\left(\frac{max(allValues) - min(allValues)}{n}\right) \quad (4)$$

- **width5bin:** Same as **width3bin** except we use 5 bins.
- **freq3bin:** Generates 3 bins of equal population size;
- **freq5bin:** Same as **freq3bin**, only this time we have *5* bins.

A.2. Nine Learners

Based on our reading of the effort estimation literature, we identified nine commonly used learners that divide into

- Two *instance-based* learners: **ABE0-1NN, ABE0-5NN**;
- Two *iterative dichotomizers*: **CART(yes),CART(no)**;
- A *neural net*: **NNet**;
- Four *regression methods*: **LReg, PCR, PLSR, SWReg**.

Instance-based learning can be used for analogy-based estimation (ABE). Since it is not practical to experiment with

the all ABE variants, we focus on two standard variants. ABE0 is our name for a very basic type of ABE that we derived from various ABE studies [**?**], [**?**], [**?**]. In **ABE0-xNN**, features are firstly normalized to 0-1 interval, then the distance between test and train instances is measured according to Euclidean distance function, x nearest neighbors are chosen from the training set and finally for finding estimated value (a.k.a adaptation procedure) the median of x nearest neighbors is calculated. We explored two different x:

- **ABE0-1NN:** Only the closest analogy is used. Since the median of a single value is itself, the estimated value in **ABE0-1NN** is the actual effort value of the closest analogy.
- **ABE0-5NN:** The 5 closest analogies are used for adaptation.

Iterative Dichotomizers seek the best attribute value $splitter$ that most simplifies the data that fall into the different splits. Each such splitter becomes a root of a tree. Sub-trees are generated by calling iterative dichotomization recursively on each of the splits. The CART iterative dichotomizer [**?**] is defined for continuous target concepts and its $splitters$ strive to reduce the GINI index of the data that falls into each split. In this study, we use two variants:

- **CART (yes):** This version prunes the generated tree using cross-validation. For each cross-validation, an internal node is made into a leaf (thus pruning its sub-nodes). The sub-tree that resulted in the lowest error rate is returned.
- **CART (no):** Uses the full tree (no pruning).

In *Neural Nets*, or **NNet**, an input layer of project details is connected to zero or more "hidden" layers which then connect to an output node (the effort prediction). The connections are weighted. If the signal arriving to a node sums to more than some threshold, the node "fires" and a weight is propagated across the network. Learning in a neural net compares the output value to the expected value, then applies some correction method to improve the edge weights (e.g. back propagation). Our **NNet** uses three layers.

This study also uses four *regression methods*. **LReg** is a simple linear regression algorithm. Given the dependent variables, this learner calculates the coefficient estimates of the independent variables. **SWreg** is the stepwise regression. As a pre-processor **SWreg** is used to select features for other learners, here we use **SWreg** as a learner (that is, the predicted value is a regression result using the features selected by the last step of **SWreg**). Partial Least Squares Regression (**PLSR**) as well as Principal Components Regression (**PCR**) are algorithms that are used to model a dependent variable. While modelling an independent variable, they both construct new independent variables as linear combinations of original independent variables. However, the ways they construct the new independent variables are different. **PCR** generates new independent variables to explain the observed variability in the actual ones. While generating new variables the dependent variable is not considered at all. In that respect, **PCR** is similar to selection of *n-many* components via **PCA** (the default value of components to select is 2 in MATLAB implementation, so we used it that way) and applying linear regression. **PLSR**, on the other hand, considers the independent variable and picks up the *n-many* of the new components (again with a default value of 2) that yield lowest error rate. Due to this particular property of **PLSR**, it usually results in a better fitting.