Optimal Project Feature Weights in Analogy-Based Cost Estimation: Improvement and Limitations

Martin Auer, *Member*, *IEEE*, Adam Trendowicz, Bernhard Graser, Ernst Haunschmid, and Stefan Biffl, *Member*, *IEEE*

Abstract—Cost estimation is a vital task in most important software project decisions such as resource allocation and bidding. Analogy-based cost estimation is particularly transparent, as it relies on historical information from similar past projects, whereby similarities are determined by comparing the projects' key attributes and features. However, one crucial aspect of the analogy-based method is not yet fully accounted for: the different impact or weighting of a project's various features. Current approaches either try to find the dominant features or require experts to weight the features. Neither of these yields optimal estimation performance. Therefore, we propose to allocate separate weights to each project feature and to find the optimal weights by extensive search. We test this approach on several real-world data sets and measure the improvements with commonly used quality metrics. We find that this method 1) increases estimation accuracy and reliability, 2) reduces the model's volatility and, thus, is likely to increase its acceptance in practice, and 3) indicates upper limits for analogy-based estimation quality as measured by standard metrics.

Index Terms—Software cost estimation, analogy-based cost estimation, project clustering, project features.

1 Introduction

Software cost estimation [1] is a vital task in software project management. It affects all major management decisions such as resource allocation, bidding, start scheduling, and risk management. Good cost estimation requires experience and judgment due to human factors (like stakeholder incentives and team dynamics) and continuous technological change in any software development environment. Formal estimation methods can provide valuable support.

Formal estimation methods measure observable project characteristics, *project features*, and derive cost estimates from these features. Typical examples of project features include the number of function points, number of user interface masks, number of developers, and the experience level of developers. In order to derive cost estimates from the chosen project features, a model relating features and costs is required. In the typical scenario of a multiproject environment, a *project portfolio*, such a model can include the features and observed costs of past projects, in order to obtain estimates in line with historical project costs. Over time, a project portfolio grows as a company takes on more new projects. When those projects are completed, their features and observed costs are added to a historical project feature database. The typical estimation procedure in a growing portfolio is thus to:

1. select and measure project features of the project to be estimated;
2. estimate the project costs using the features observed and the model derived from past completed projects; and
3. upon a project's completion, add the project's features and final costs to the historical feature database and recalibrate the model based on this new, larger database.

For any cost estimation method to be useful, it should both produce sound estimates and be accepted and trusted by the practitioner. One method that meets these conditions is *analogy-based cost estimation* [2]. The analogy-based approach is reported to perform well with respect to estimation accuracy when compared to other estimation methods [3]. In addition, it is by nature transparent (as opposed to, for example, black-box approaches like neural networks) and, therefore, likely to be accepted by practitioners [4].

It works as follows: Given a new project to be estimated along with its features, analogy-based cost estimation tries to find projects with similar features in the historical feature database. Those projects' costs are then used to create the new estimate. For example, consider estimating a new project with two features: "number of developers" and

- M. Auer is with the Institute of Software Technology and Interactive Systems, Vienna University of Technology, and can be reached at 145 W. 58 St., 10019 New York, NY. E-mail: martin.auer@tuwien.ac.at.
- A. Trendowicz is with the Fraunhofer Institute for Experimental Software Engineering, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany. E-mail: adam.trendowicz@iese.fhg.de.
- B. Graser is with Act Management Consulting, Haslingergasse 55/5/17, A-1160 Vienna, Austria. E-mail: bernhard.graser@gwd.at.
- E. Haunschmid is with the Computing Services Department, Vienna University of Technology, Wiedner Hauptstrasse 8-10, A-1040 Vienna, Austria. E-mail: haunschmid@zid.tuwien.ac.at.
- S. Biffl is with the Institute of Software Technology and Interactive Systems, Vienna University of Technology, Favoritenstr. 9-11/188, A-1040 Vienna, Austria. E-mail: stefan.biffl@tuwien.ac.at.

Manuscript received 13 Aug. 2004; revised 10 Oct. 2005; accepted 6 Jan. 2006; published online ? Feb., 2006.
Recommended for acceptance by J. Offutt.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-0171-0804.

"number of user interface masks." There may be several projects in the historical feature database with a similar number of developers and interface masks (given a sufficient number of historical projects). The average of those similar projects' known costs is the new estimate.

However, conventional analogy-based cost estimation methods are inadequate when dealing with individual project features, as it is reasonable to assume that some features should be more influential than others. Yet often, this issue is either not addressed [5], or handled by merely trying to identify the most dominant project features [2].

Therefore, this paper proposes an improved analogy-based cost estimation algorithm. Our approach uses extensive search to optimally weight individual project features, in order to find a more realistic measure of project similarity. In this way, some features become more influential in determining project similarities. By extending earlier attempts presented in [6], this paper:

- describes an extensive search algorithm for optimal project feature weights,
- applies the approach to several real-world portfolio data sets and compares it to a conventional analogy-based approach, and
- quantifies the obtained quality improvements.

Our approach produces better estimates, as measured by commonly used estimation accuracy and reliability metrics. Furthermore, it is acceptable to practitioners in environments that allow for consistent collection of project features: It keeps the transparency of the conventional analogy-based approach, but adapts more flexibly to real-world project features of varying importance.

The proposed approach requires extensive computation for model calibration. However, this takes place only upon project completion, i.e., at step 3 of the estimation procedure. At that time, a completed project's features and observed costs are stored in the historical feature database, and the model is recalibrated. At the time of estimation, however, no substantial computation is required—the estimate is given in realtime.

Section 2 presents an overview of related work. Section 3 describes the algorithm and how to deal with high-dimensional portfolio data sets that require extensive computation. Section 4 gives the data sets used to analyze the proposed approach. Section 5 presents the results, and Section 6 discusses the implications regarding estimation quality, model volatility, and estimation quality barriers. Section 7 concludes the paper and provides directions for further research.

2 Related Work

There are various formal methods that have been proposed to support software cost estimation, for example, COCOMO [7], neural networks [8], pattern recognition [9], and expert judgment [10].

Most methods rely on the same assumption: Similar projects are likely to have similar cost characteristics. This is implemented in the most straightforward way by the analogy-based estimation method [11], [12]. Wieczorek [3] compares several cost estimation approaches and the analogy-based method ranks among the best. Shepperd and Schofield [2] claim that analogy-based estimation outperforms regression techniques for several real-world data sets. Myrtveit and Stensrud [13] compare the analogy approach to regression methods. Their results indicate that the analogy-based method may not be the best choice in some environments if only estimation accuracy is considered. However, there are other equally important criteria to consider when selecting an estimation method, especially the method's transparency and its acceptance by practitioners (as noted by Mendes et al. [14]). In these cases, the analogy method is considered to be preferable to black-box approaches such as neural networks (see, for example, Finnie and Wittig [4]).

When estimating a new project, the analogy-based method consists of several steps. First, measure the observable features of the new project at the time of estimation. Then, identify projects with similar feature values from the historical feature database. Finally, determine the new estimate using the known costs of the chosen historical projects.

More precisely, a project p is described by a list of features $\langle e, d_1, \ldots, d_l \rangle$, where $d_1, \ldots, d_l$ denote those features that are observed at the time of estimation, and e denotes the project's costs upon completion [15] (because the features' value ranges differ, they are first scaled to the interval $[0, 1]$). The similarity of two projects can be defined as a weighted Euclidean distance δ over the features $d_1, \ldots, d_l$:

$$\delta(p, p') = \sqrt{\sum_{i=1}^{l} w_i (d_i - d'_i)^2}. \qquad (1)$$

A small distance indicates a high degree of similarity. When a new project is estimated, its distances to each project in the historical feature database are calculated. The costs of the most similar projects then determine the new cost estimate.

A fundamental question in this model is how to set the feature weights w_i since individual features should influence project similarity to a different degree. Various approaches have been proposed:

- Set all project feature weights to identical values: $w_i = 1, i = 1, \ldots, l$ [5].
- Set each project feature weight to a value determined by human judgment [16].
- Set each project feature weight to a value obtained by statistical analysis, such as inverse variance or range values [12], or to values that depend on the correlation of a feature with historical costs (for example, $w_i = 1$ or $w_i = 2$, depending on a threshold correlation) [5].
- Set each project feature weight to either 0 or 1 so that an estimation quality metric is maximized. This brute-force approach proposed by Shepperd and Schofield [2] tries to identify a subset of important features. Once these features are identified, they are all given the same weight.

Treating all feature weights the same is not optimal, as some features influence project costs more than others; this is supported by Mendes et al. [5]. Experts, on the other hand, may be reluctant to set the weights manually due to the additional effort required to analyze and set the weights.

In turn, it remains difficult to agree on the correct statistical analysis and on the corresponding feature weight values (for example, the proposed values "1" and "2" seem rather arbitrary). In addition, this approach makes the model less transparent and acceptable in real-world environments—eliminating a key advantage of analogy-based cost estimation [4].

The last approach—searching for a subset of important features—fails to account for each feature's individual influence on project similarity. Also, the resulting feature weights are highly volatile over the lifetime of a growing portfolio, which may make the model less acceptable.

This paper extends Shepperd et al.'s feature subset approach [11]. It proposes to use flexible weights to account for the individual influences of project features without sacrificing the transparency and simplicity of analogy-based estimation. The paper also tests the new approach on real-world project feature sets, and compares its estimation quality to that of Shepperd and Schofield's approach.

3 Extensive Feature Weight Search

This section describes an algorithm to determine optimal feature weights with respect to estimation quality metrics and explains how to handle high-dimensional data sets that require extensive computation.

3.1 Method and Metrics

In analogy-based estimation, a new project's cost is estimated by identifying the most similar past projects—i.e., projects with similar feature values—and taking their mean observed costs as the new estimate.

A completed project p, member of a portfolio P, can be described by values $\langle e_p, d_{p,1}, \ldots, d_{p,l} \rangle$, where e_p denotes the final cost observed upon project completion, and $d_{p,1}, \ldots, d_{p,l}$ denote l feature values observed at the project's time of estimation. For a new project p^*, whose costs are to be estimated, only the $d_{p^*,i}$ are known. To create an estimate $\hat{e}_{p^*}$ for the new project's costs, the weighted Euclidean distance δ (see Section 2) over the known features $d_{-,i}$ is first used to determine the set Q_{p^*} of projects that are similar to p^*, i.e., those with a minimal distance:

$$Q_{p^*} = \{p \in P \setminus \{p^*\} | \delta(p, p^*) = min_{p' \in P \setminus \{p^*\}} \delta(p', p^*)\}.$$

The cost estimate $\hat{e}_{p^*}$ for the new project p^* is then given by the average costs of the projects in that set: $\hat{e}_{p^*} = |Q_{p^*}|^{-1} \sum_{p \in Q_{p^*}} e_p$.[1]

Several estimation quality metrics can be used to assess the performance of estimation methods. Commonly used metrics are the "mean magnitude of relative error" ($MMRE$), and the "percentage of predictions that fall within 25 percent of the actual value" ($Pred_{25}$). Although their application has been criticized, they remain widely used and are considered to have no simple generic replacement (see, for example, Foss et al. [17]).

These metrics are obtained in a so-called jackknifing approach: For a given portfolio P, a completed project with known final costs is treated as if its costs were still to be estimated, and the remaining projects in the portfolio are used to create that estimate. This is repeated for every project in the portfolio, and the obtained relative errors are aggregated. Formally, for each $p \in P$, the estimate $\hat{e}_p$ is calculated, yielding the relative error $r_p = (\hat{e}_p - e_p)/e_p$. The quality metrics are then defined as follows:

$$MMRE = |P|^{-1} \sum_{p \in P} |r_p|, \tag{2}$$

$$Pred_{25} = |P|^{-1} |\{p \in P \Big| |r_p| \leq .25\}|. \tag{3}$$

In addition, Var_r, the variance of the relative error, can be used as a measure of the estimation method's reliability. The choice of a particular metric depends on the preference of the estimator: Large estimation outliers, for example, influence the average-based $MMRE$ more than $Pred_{25}$.

These quality metrics can be used to calibrate an estimation model. In our case, the model parameters are the weights $w_1, \ldots, w_l$ that are associated with the l features in the weighted Euclidean distance function δ. As previously outlined, these weights are usually either set manually by experts, or—in the case of the tool ArchAngel [2]—they are set to 0 or 1 in a brute-force search for a subset of important features. Yet, these methods provide only crude approximations for the features' individual weights.

This paper proposes to extensively search for weights that maximize certain standard estimation quality metrics. Each weight takes a value in the interval $[0, 1]$. The weights $\langle w_1, \ldots, w_l \rangle$ are called optimal with respect to a quality measure (for example, $MMRE$-optimal) if, for a given portfolio P, they yield the best quality metric value of all examined combinations of weight values. This corresponds to a minimum value in the case of $MMRE$, or to a maximum in the case of $Pred_{25}$.

Project portfolios grow as new projects are started. We use the term "portfolio growth stage" to denote the intermediate portfolios obtained by adding one new project. The optimal feature weights can be different at each such growth stage, as more and more projects are completed and their cost information becomes available for model calibration. If only two possible weight values are used (as in the approach of Shepperd and Schofield [2]), changes in optimal feature weights appear very discontinuous: Weights are turned from 1 to 0 or vice versa. This makes the model's overall behavior appear highly volatile and its estimates untrustworthy. A higher number of possible weight values should reduce this volatility.

Thus, a volatility measure is used to determine the impact of extensive weight search on weight volatility. If P_t indicates an intermediate portfolio stage of t projects (t ranges from initial size s_{min} to total number of projects n) and, if $\langle w_1^t, \ldots, w_l^t \rangle$ denote the optimal weights for each P_t,

[1] The set often contains only one project, especially if some project features are not categorical features. By modifying the strict equality condition for Q_{p^*}, larger sets can be selected.

then the mean feature weight volatility ($MFWV$) of the portfolio P is:

$$MFWV = \frac{\sum_{i=s_{min}+1}^{n} \sum_{j=1}^{l} |w_l^i - w_l^{i-1}|}{l(n - s_{min})}. \quad (4)$$

This metric describes the mean absolute change of weights over the lifetime of a portfolio.

3.2 Implementation

The proposed approach allocates separate weights w_i to the l project features d_i. If w denotes the number of possible weight values, then each weight w_i can be set to a value from the set $\{0, 1/(w-1), 2/(w-1), \ldots, (w-2)/(w-1), 1\}$. For example, with $w = 3$, each project feature can be assigned a weight of either 0, $.5$, or 1. To search for the optimal weight combination for a given portfolio, w^l combinations of weights must be analyzed. We analyze the cases $w = 2, 3, 5, 7, 9$: $w = 2$ corresponds to the traditional approach, while the larger, odd values for w allow for the feature weight value .5.

In a real-world portfolio that grows over time, model calibration takes place whenever a project is completed and its features and observed costs are stored in the historical feature database. To analyze the improvements of our approach over the lifetime of a portfolio, the basic algorithm is therefore applied at each stage of a growing portfolio:

1. Given a portfolio $P = \{p_1, \ldots, p_n\}$, choose an initial portfolio size s_{min} and the number of possible weights w, set the current portfolio stage to $P_t = \{p_1, \ldots, p_{s_{min}}\}$, and select a quality metric (for example, $MMRE$).
2. Determine the optimal weight values for the current portfolio P_t with respect to the chosen quality metric by extensive search. To cross-check the improvement, determine additional quality metrics, for example, $Pred_{25}$ or Var_r, for this weight configuration. Note, the weights may not be optimal with respect to these additional metrics.
3. Go to the next portfolio growth stage—while there are projects left in the portfolio, add one project to the current portfolio (i.e., increase t), and repeat Steps 2 and 3.

The computational effort required is proportional to w^l. If the algorithm is computationally feasible (i.e., computable on a given system within a specified amount of computation time) for some number of possible weight values w, its computation time T can be observed and used to predict the computation time for larger values of w. For example, to increase w by 1, a computation time of $Tw^{-l}(w+1)^l$ would be required.

If this prediction exceeds the available computational resources, a simple modification to the above algorithm gets rid of features that seem to be less relevant; features are dropped until the expected computation time becomes feasible. We propose to drop those features whose weights are 0 during all stages of the growing portfolio, for the previous, smaller value of w. If there are none, we drop those whose weights were 0 during at least 90 percent of the portfolio stages. While it could be that a higher number of possible weight values "activates" some of those features, they can be considered less important than other features (much like covariates with small coefficients in regression-based models). If several features qualify, the one with the highest index is dropped.

Consider the following example: Limited computational resources make it necessary to drop two features in order to increase w from 4 to 5. For $w \leq 4$, the weight w_3 of feature d_3 is zero in all portfolio stages, and the weights w_5 and w_7 of features d_5 and d_7 are zero for all but one stage. Then, features d_3 and d_7 are dropped before increasing w.

Practitioners might be reluctant to apply methods that require such high computational effort. This approach, however, is still usable for the following reasons:

- First, and most importantly, the effort is required at model calibration, not at the time of estimation. At the time of estimation, the estimates are given in realtime. The expensive recalibration takes place only when a project is completed and its features and observed costs are stored to the historical feature database. This can even be implemented transparently, for example, by a stored procedure that is triggered automatically when new project information is stored.
- The method is even acceptable if for some reason model recalibration is required during the estimation. Any real-world cost estimation process takes place over several days, usually involving the feedback of several team members, procedures such as top-down or bottom-up estimation, and time-consuming budget politics. Thus, during this procedure, it is easily possible to apply time-intensive algorithms (typically, they are scheduled to run overnight). Similar approaches are, for example, very common in price and risk analysis applications in financial engineering.
- In addition, the algorithm can easily be parallelized even on low-cost hardware. The data set characteristics are also often favorable: The data sets usually contain only few project entries, typically between 20 and 100, and the number of features is usually small (6 to 18 in the analyzed data sets).

The algorithm is implemented as a Java tool, AmberAngel. Unlike other analogy-based estimation tools, it is a command line tool specifically designed to operate in batch mode; its output is suited for further processing in spreadsheet applications. By setting the number of possible feature weight values to 2, the conventional feature subset approach is simulated (this produces the same results as Shepperd and Schofield's tool). The computation is performed on one processor of a Dual AMD Opteron 2GHz Linux system.

4 Data Sets and Analysis Procedure

We apply the proposed approach to several real-world portfolio data sets. Some sets are available in the public domain [18], [19], [20]. In addition, our partners at the Fraunhofer Institute were able to get access to several data sets not published in detail, the Esa and Laturi data sets.

TABLE 1
Portfolio Data Sets

Data set	#Features, l	#Projects, n
Albrecht24 [18]	6	24
Desharnais44 [19]	7	44
Desharnais23 [19]	7	23
Desharnais10 [19]	7	10
Esa29 [3]	11	29
Esa13 [3]	11	13
Kemerer15 [20]	3	15
Laturi12 [3]	18	12
Laturi11 [3]	18	11
Laturi10a [3]	18	10
Laturi10b [3]	18	10

The former data set contains multiorganizational data from the European Space Agency, in the domains of space, military, and business applications; it is administered by the international business school INSEAD. The latter data set contains data from the Laturi project, a collection of project metrics from several companies in Finland in the domain of business application systems; the database is administered by the Software Technology Transfer Finland Ltd.

Many authors point out that portfolio data sets are most valuable if used to estimate within the very same company or development environment [21], [22]. Therefore, the larger data sets were broken into company-specific ones, yielding 11 data sets. Table 1 gives an overview on the number of features and number of projects of these portfolios.

Table 2 lists the features of the first projects of the Albrecht data set. The first column, "name," is the project identifier; it is not used in the calculation. The second column, "effort," is the estimation target. The remaining six columns are the six project features describing this portfolio's projects: lines of code (sloc), number of function points (fp), file handles (file), and masks for input (in), output (out), and inquiries (inq). Note that analogy-based cost estimation does not depend on certain types of features, it just operates on quantitative and (in our case) categorical data. Please refer to [3] for more details about the data sets used.

Some minor preliminary preparations are made to the data sets:

- One column (usually, "effort") is chosen as the estimation target; additional effort-related measures like productivity or duration are dropped.
- Categorical features (for example, "type of programming language") are denoted so that our tool is able to apply a special distance function to such features (0 for same, 1 for different categories).
- No projects are removed from the data sets, even if they appear to be outliers.

In Section 3, a modification of the extensive search for optimal weights is described, which deals with portfolios containing a large number of features. Finding optimal weights for these portfolios may require extensive computation, because the effort is proportional to w^l. To be able to increase w and to find weights within a given maximum computation time, we propose to decrease l by removing features that are 0 in all or most portfolio stages analyzed with a lower w.

TABLE 2
Albrecht Data Set (Partial)

name	effort	in	out	file	inq	fp	sloc
1	10 240	25	150	60	75	1 750	130
2	10 520	193	98	36	70	1 902	318
3	1 110	70	27	12	0	428	20
4	2 110	40	60	12	20	759	54
5	2 880	10	69	9	1	431	62
6	1 000	13	19	23	0	283	28
...							

We use a maximum computation time of 24 hours. Removing some features is then necessary for some of the Esa and Laturi data sets. For example, for data set Esa13, one feature has to be removed at $w = 6$, $w = 7$, and $w = 9$, respectively, to make these w values computationally feasible. Here, $w = 10$ is the maximum number of possible weight values; no more features can be removed under the criteria described in Section 3. The data sets Lat12 and Lat11 are analyzed up to $w = 5$ possible weight values; no more features can be removed under the used criteria.

5 Results

Fig. 1 shows the improvement of the extensive feature weight search versus the feature subset approach proposed by Shepperd and Schofield [2]. Each portfolio is analyzed at all stages of the growing portfolio ($s_{min} = 5$). The solid line gives the improvement, i.e., the reduction, of $MMRE$ in percent at each stage. The dashed line gives the improvement, i.e., the increase, of $Pred_{25}$. Finally, the dotted line gives the improvement, i.e., the reduction, of Var_r.

Fig. 1 also gives the number of possible weight values w for each portfolio. It varies due to the different computational requirements of the various portfolios: Some portfolios have more features, and some have more features that can be removed.

Table 3 summarizes the same data and gives the average improvements of the three estimation quality metrics $MMRE$, $Pred_{25}$, and Var_r over all intermediate stages of a growing portfolio.

Fig. 2 shows the impact of different numbers of possible weight values in the case of the Albrecht data set. The box plots show the distributions of the improvements in all intermediate stages of the growing portfolio, for two (conventional approach, i.e., no improvement), three, five, seven, and nine possible weight values, respectively.

Table 3 gives the improvement of the volatility measure $MFWV$ in percent. There are $n - s_{min} + 1$ $MMRE$ values in

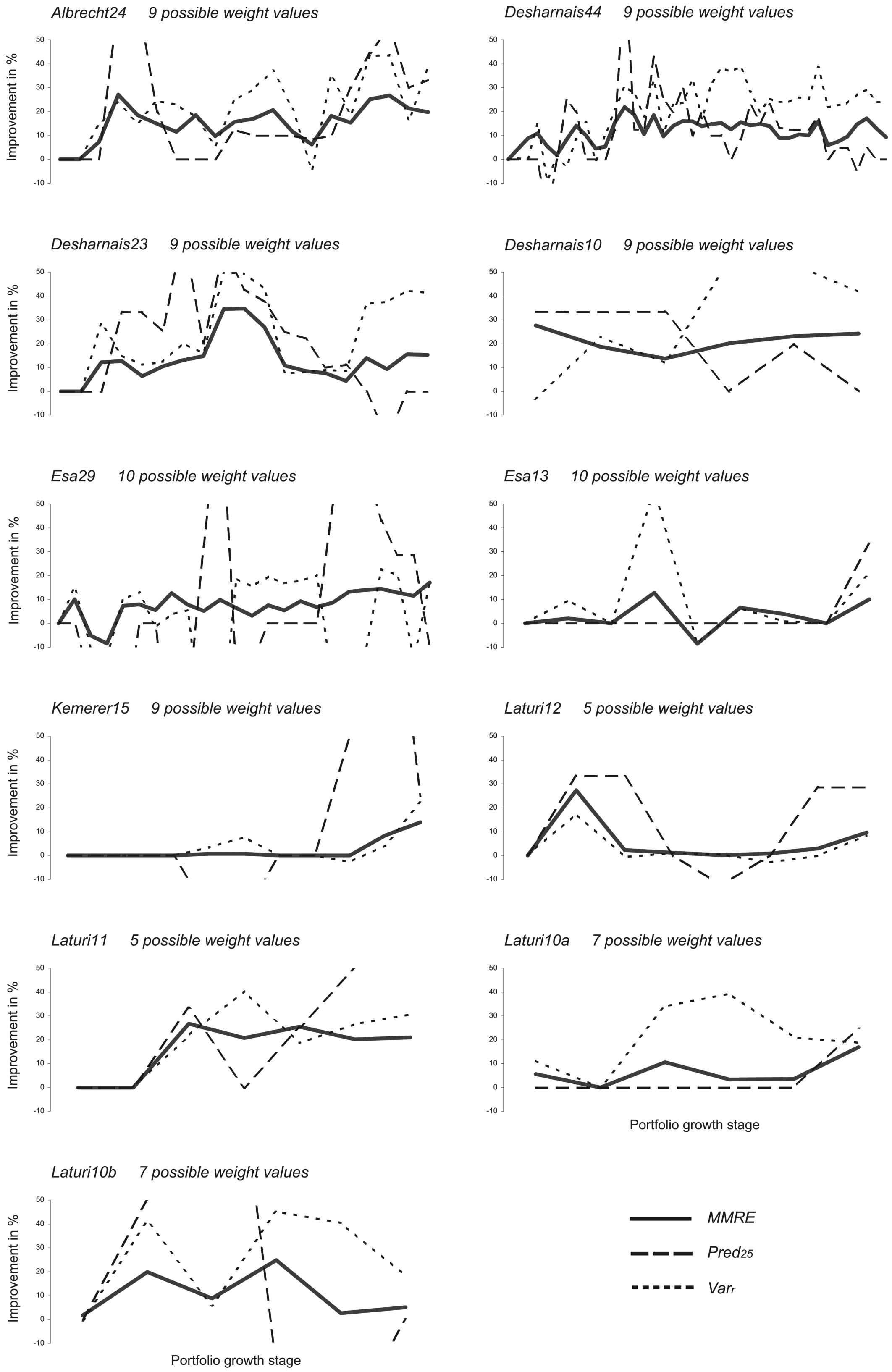


Fig. 1. $MMRE$, $Pred_{25}$, and Var_r improvement in percent for all data sets and portfolio growth stages.

a growing portfolio, but there is just one $MFWV$ value for an entire growing portfolio, as it is derived from the absolute changes of the $MMRE$-optimal weights at all intermediate stages.

To better illustrate the concept of weight volatility, Fig. 3 shows how, in a growing portfolio, the higher number of possible weight values affects the weights' volatility. It shows the values of the $MMRE$-optimal weights at each

TABLE 3
Mean $MMRE$, $Pred_{25}$, and Var_r Improvement in Percent; $MFWV$ Improvement in Percent

Data set	$MMRE$	$Pred_{25}$	Var_r	$MFWV$
Albrecht24 [18]	15	20	21	34
Desharnais44 [19]	12	11	20	43
Desharnais23 [19]	13	19	23	65
Desharnais10 [19]	21	20	30	18
Esa29 [3]	8	7	1	58
Esa13 [3]	3	4	9	29
Kemerer15 [20]	2	16	3	34
Laturi12 [3]	6	14	3	8
Laturi11 [3]	16	25	20	27
Laturi10a [3]	7	4	21	36
Laturi10b [3]	11	30	25	10

intermediate stage of the growing Desharnais23 portfolio (read the diagram from right to left, as more projects are being added to the initial portfolio). In this case, there are seven features. The upper part shows the behavior of the $MMRE$-optimal feature weights if only $w = 2$ weight values are allowed (0 or 1); the lower part depicts the case of $w = 9$ possible weight values.

6 Discussion

Our proposed method extends Shepperd and Schofield's feature subset approach by searching for optimal feature weights. The two approaches are compared using estimation quality metrics on several real-world data sets. (Note: to keep the tests independent of human judgment, outliers are not removed from the data sets.) We found three main results:

- *Accuracy and reliability is improved.* Our approach determines optimal feature weights by minimizing a standard error metric, $MMRE$, at all stages of a growing portfolio. Over these intermediate portfolio stages, the $MMRE$ is improved, i.e., reduced, by 2 to 21 percent on average for all analyzed data sets.

  Optimizing a model with respect to one quality metric may adversely affect other metrics. For example, accuracy (measured by $MMRE$) may be optimized at the expense of reliability (as measured by Var_r). Therefore, two other estimation quality metrics—$Pred_{25}$ and Var_r—are also analyzed.

  The $Pred_{25}$-value oscillates in the growing portfolios and, in some instances, performs worse. However, over all growing portfolios, the average $Pred_{25}$-value is improved (see Fig. 1). The same holds for Var_r, a measure of estimation reliability. Thus, on average, both accuracy and reliability are improved in all tested data sets, as measured by different metrics.

  The average improvement depends on the initial portfolio size. This is best illustrated by the Kemerer data set. In the final stages of its growing portfolio, the $MMRE$ is improved by 10 percent or more. However, since there is no improvement in the initial stages of the growing portfolio, the average $MMRE$ improvement is just 2 percent.

  Another factor to consider is the number of possible weight values. For data sets with many features, fewer weight values are used due to computational constraints. Some high-dimensional data sets, such as Laturi12, are analyzed with a maximum of five possible weight values. Other sets, such as Esa29, can be analyzed using up to 10 possible weight values. Fig. 2 shows how an increasing number of weight values affects the distribution of $MMRE$ improvements in a growing portfolio. While using more feature weight values is

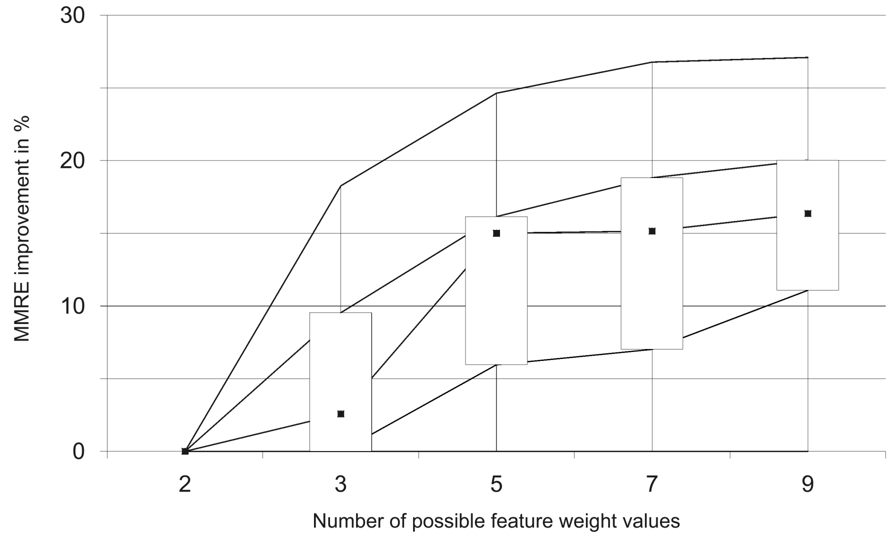
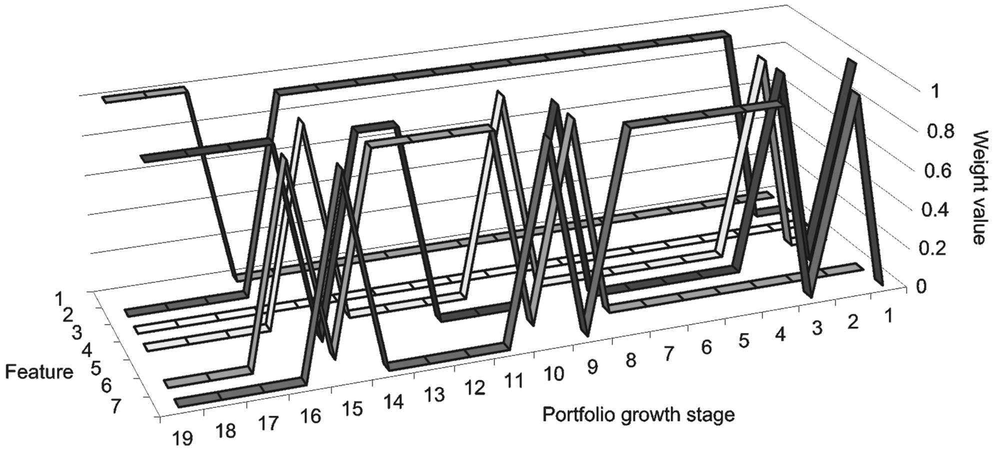
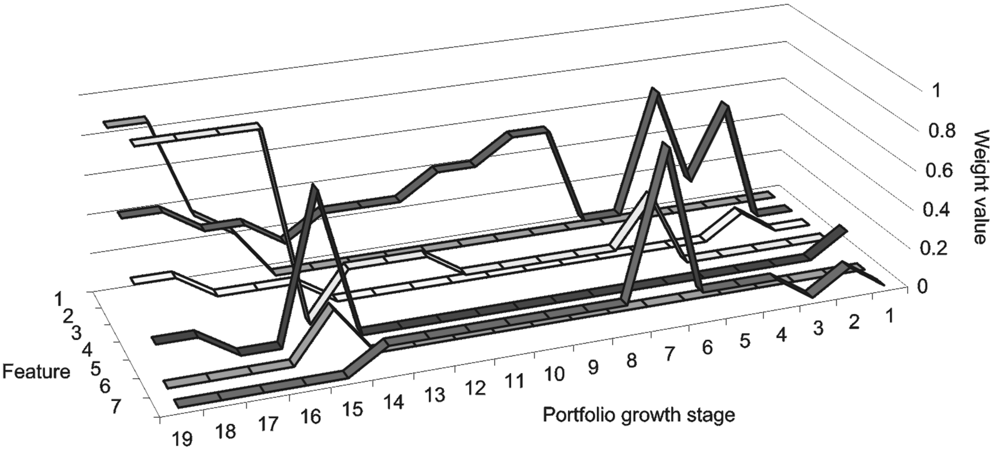


Fig. 2. Distributions of $MMRE$ improvement over all growth stages of the Albrecht portfolio, for different number of possible weight values (the traditional approach uses two values).



(a)



(b)

Fig. 3. Weight values for the seven features in the growing Desharnais23 portfolio—two versus nine possible weight values.

better, the improvements typically level off after five or seven possible weight values.

For high-dimensional data sets, the new algorithm removes features that appear unimportant, in order to make an increased number of weight values computationally feasible (here, those features with a weight value of 0 in at least 90 percent of portfolio stages are removed). As stated before, the improvements gained by increasing the number of weight values levels off. As a result, the new approach performs worse in some rare instances. One example is Esa29 (Fig. 1), where the $MMRE$ actually deteriorates at portfolio stages $n = 7$ and $n = 8$.

- *Volatility is improved*. A measure for the average changes of feature weights in the case of a growing portfolio is $MFWV$. In all data sets, this measure is reduced by 8 percent to 65 percent. Fig. 3 illustrates this well: While the weights oscillate in a seemingly random way in the case of two possible weight values, the higher number of nine possible weight values yields a more continuous behavior over the lifetime of the growing portfolio. For example, consider weight w_7: For two possible weight values, it changes five times from 0 to 1 or from 1 to 0 between the 8th and 16th project added; yet it stays very stable during the same portfolio stages in case of nine possible weight values.

  This indicates that feature weights are related directly to the estimation quality and are not just arbitrary parameters determined during the optimization procedure. We expect the less volatile behavior of the feature weights over the portfolio lifetime to inspire more confidence in the estimation model, leading to greater acceptance by practitioners.

The improvements depend on a data set's characteristics, such as the number of outliers and the number of features, and in our case are around 10 percent if measured by $MMRE$. These improvements also provide a limit to analogy-based estimation quality:

- *Obtained estimates represent upper limits of analogy-based estimation quality as measured by standard metrics*. Analogy-based cost estimation relies on several assumptions: Euclidean distance is used, most similar projects determine the Estimate, and $MMRE$ is a reasonable quality metric. Under these basic assumptions, the proposed approach yields optimal feature weights—not even expert calibration can determine weight values that result in better quality metrics.

It is worthwhile to remember that in the real world, formal methods such as the proposed approach are just supporting tools for human estimation. However, human estimation uses the same data and applies a similar logic to the analogy-based method—that of relating unknown outcomes to known comparable experiences. Therefore, the absolute performance of the proposed approach can indicate whether human estimation is likely to perform well in a given environment.

This approach can be refined in several ways, especially when it comes to the handling of outliers. However, at least the question of which feature weight values to use can be answered in a formal way that is both transparent and independent of human judgment.

7 Conclusion and Outlook

The method proposed in this paper uses extensive search to find optimal project feature weights for analogy-based cost estimation. In this approach, every feature has a distinct influence on the search for similar projects in a historical feature database. It eliminates the need for experts to set the weights manually, based on their own experience. For high-dimensional data sets, a simple procedure is proposed to eliminate those features that are unlikely to influence the estimation quality. The approach is tested on several real-world portfolio data sets and compared to the feature subset approach proposed by Shepperd and Schofield [2].

The new approach outperforms existing methods with respect to commonly used estimation quality metrics. It results in a less volatile and, thus, more acceptable, behavior of the feature weights over the typical lifetime of a growing portfolio. Also, the achieved estimation quality represents an upper limit to estimation quality under the typical assumptions of the analogy approach—even experts can not allocate weights that yield better metrics. Although model calibration is computationally expensive, this only takes place when the historical feature database is updated —at the critical time of estimation, estimates are obtained in realtime.

Our future research will deal with outliers in portfolio data sets by analyzing more industrial data. We especially look forward to analyzing more recent project feature sets from different application environments; a possible target is the growing number of open-source projects with standardized project measurement procedures. Furthermore, since real-world application of the proposed method requires an easy-to-access user interface, one goal is to provide a spreadsheet plug-in to perform the model calibration seamlessly. Another aim is to extend the approach to deal with more fine-grained project feature sets, allowing for clustering of project components.

Acknowledgments

The authors would like to thank the Information Technology Services of the Vienna University of Technology for providing computational resources and Isabella Wieczorek for supplying some portfolio data sets. Finally, thanks to the enchanting Jean Wang and to Henry Sanderson for their kind assistance in drafting the document.

References

[1] C. Jones, *Estimating Software Costs.* McGraw-Hill, 1998.

[2] M. Shepperd and C. Schofield, "Estimating Software Project Effort Using Analogies," *IEEE Trans. Software Eng.,* vol. 23, no. 12, pp. 736-743, Nov. 1997.

[3] I. Wieczorek, "Improved Software Cost Estimation—A Robust and Interpretable Modeling Method and a Comprehensive Empirical Investigation," PhD dissertation, Fraunhofer Inst. Für Experimentelles Software Eng., 2001.

[4] G. Finnie and G. Wittig, "A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case Based Reasoning and Regression Models," *J. Systems Software,* vol. 39, pp. 281-289, 1997.

[5] E. Mendes, I. Watson, C. Triggs, N. Mosley, and S. Counsell, "A Comparative Study of Cost Estimation Models for Web Hypermedia Applications," *Empirical Software Eng.,* vol. 8, pp. 163-196, 2003.

[6] M. Auer and S. Biffl, "Increasing the Accuracy and Reliability of Analogy-Based Cost Estimation Techniques with Extensive Project Feature Dimension Weighting," *Proc. ACM-IEEE Int'l Symp. Empirical Software Eng. (ISESE '04),* Aug. 2004.

[7] B. Boehm, E. Horowitz, R. Madachy, D. Reifer, B. Clark, B. Steece, A. Brown, S. Chulani, and C. Abts, *Software Cost Estimation with Cocomo II.* Prentice Hall, 2000.

[8] J. Bode, "Decision Support with Neural Networks in the Management of Research and Development: Concepts and Application to Cost Estimation," *Information and Management,* no. 34, pp. 33-40, 1998.

[9] C. Briand and V.R. Basili, "A Pattern Recognition Approach for Software Engineering Data Analysis," *IEEE Trans. Software Eng.,* vol. 18, no. 11, pp. 931-942, 1992.

[10] R. Hughes, "Expert Judgement as an Estimating Method," *Information and Software Technology,* vol. 38, no. 2, pp. 67-75, 1996.

[11] M. Shepperd, C. Schofield, and B. Kitchenham, "Effort Estimation Using Analogy," *Proc. 18th Int'l Conf. Software Eng. (ICSE '96),* pp. 170-178, Mar. 1996.

[12] L. Angelis and I. Stamelos, "A Simulation Tool for Efficient Analogy Based Cost Estimation," *Empirical Software Eng.,* vol. 5, pp. 35-68, 2000.

[13] I. Myrtveit and E. Stensrud, "A Controlled Experiment to Assess the Benefits of Estimating with Analogy and Regression Models," *IEEE Trans. Software Eng.,* vol. 25, no. 4, pp. 510-525, July/Aug. 1999.

[14] E. Mendes, N. Mosley, and S. Counsell, "Do Adaptation Rules Improve Web Cost Estimation?" *Proc. 14th ACM Conf. Hypertext and Hypermedia (HYPERTEXT '03),* Aug. 2003.

[15] M. Auer, B. Graser, and S. Biffl, "An Approach to Visualizing Empirical Software Project Portfolio Data Using Multidimensional Scaling," *Proc. IEEE Int'l Conf. Information Reuse and Integration (IRI '03),* Oct. 2003.

[16] F. Walkerden and R. Jeffery, "An Empirical Study of Analogy-Based Software Effort Estimation," *Empirical Software Eng.,* vol. 4, pp. 135-158, 1999.

[17] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit, "A Simulation Study of the Model Evaluation Criterion MMRE," *IEEE Trans. Software Eng.,* vol. 29, no. 11, pp. 985-995, 2003.

[18] A. Albrecht and S. Gaffney, "Software Function, Source Lines of Code and Development Effort Prediction: A Software Science Validation," *IEEE Trans. Software Eng.,* vol. 9, no. 6, pp. 639-648, 1983.

[19] J. Desharnais, "Analyse Statistique de la Productivitie des Projets Informatique a Partie de la Technique des Point des Fonction," Master's thesis, Univ. of Montreal, 1989.

[20] C. Kemerer, "An Empirical Validation of Software Cost Estimation Models," *Comm. ACM,* pp. 416-429, May 1987.

[21] I. Wieczorek and M. Ruhe, "How Valuable is Company-Specific Data Compared to Multicompany Data for Software Cost Estimation?" *Proc. Eighth Int'l Symp. Software Metrics (METRICS '02)*, pp. 237-248, June 2002.

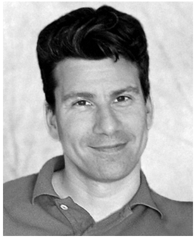
[22] K. Maxwell and L.v. Wassenhove, "Software Development Productivity of European Space, Military, and Industrial Applications," *IEEE Trans. Software Eng.*, vol. 22, no. 10, pp. 706-718, 1996.

Martin Auer holds degrees in mathematics from the University of Vienna and computer science from the Vienna University of Technology, a PhD degree in computer science from the Vienna University of Technology, and an MA degree in mathematics of finance from Columbia University, New York. He is an associate at Bank of America, New York, in the area of asset-backed securities. He was a lecturer at the Vienna University of Technology, a management information system consultant at Raiffeisen Central Bank, Austria, and a software engineer at Debis, Austria. He has published in the areas of software cost estimation and high-performance computing. He is a member of the IEEE.

Adam Trendowicz received the BSc degree in computer science and the MSc degree in software engineering from the Poznan University of Technology, Poland, in 2000. He is currently a researcher at the Fraunhofer Institute for Experimental Software Engineering (IESE), Kaiserslautern, Germany, in the Processes and Measurement Department. Before that, he worked as a software engineering consultant at Q-Labs GmbH, Germany. His research and industrial activities include software cost modeling, measurement, and data analysis.

Bernhard Graser received the degree in computer science from the Vienna University of Technology. He is a management consultant at Act Management Consulting (former KPMG consulting). His fields of interest include requirements engineering, IT governance, IT project management, and IT controlling. He was a lecturer at the Vienna University of Technology and has published in the area of software cost estimation.

Ernst Haunschmid received the degree in mathematics and the PhD degree in mathematics from the Vienna University of Technology. He published in the areas of high-performance computing and numerical linear algebra.



Stefan Biffl received the MS and PhD degrees in computer science from the Vienna University of Technology and the MS degree in social and economic sciences from the University of Vienna. He is an associate professor of software engineering at the Vienna University of Technology. His research interests include project and quality management in software engineering, with a special focus on value-based software engineering and empirical evaluation. He is a member of the ACM, the IEEE, the Austrian Computer Society, and the IFIP Technical Committee on Software Engineering.