Predicting Software Project Effort: A Grey Relational Analysis Based Method

Qinbao Song[1] and Martin Shepperd[2]

[1]Xi'an Jiaotong University, China

qbsong@mail.xjtu.edu.cn

[2]Brunel University, UK

martin.shepperd@brunel.ac.uk

December 11, 2010

Abstract

The inherent uncertainty of the software development process presents particular challenges for software effort prediction. We need to systematically address missing data values, outlier detection, feature subset selection and the continuous evolution of predictions as the project unfolds, and all of this in the context of data-starvation and noisy data. However, in this paper, we particularly focus on outlier detection, feature subset selection, and effort prediction at an early stage of a project. We propose a novel approach of using Grey Relational Analysis (GRA) from Grey System Theory (GST), which is a recently developed system engineering theory based on the uncertainty of small samples. In this work we address some of the theoretical challenges in applying GRA to outlier detection, feature subset selection, and effort prediction, and then evaluate our approach on five publicly available industrial data sets using both stepwise regression and Analogy as benchmarks. The results are very encouraging in the sense of being comparable or better than other machine learning techniques and thus indicate that the method has considerable potential.

Keywords: software project estimation, effort prediction, feature subset selection, outlier detection, Grey Relational Analysis.

1 Introduction

Software development, or our understanding of it, is a gradual and evolving process; we know less at the beginning than at the completion of a project. As the software development proceeds, we gather and analyze incomplete information and try to predict the remaining development process. But each prediction is unlikely to be unique, it is modified or changed as more information becomes available. This means that effort prediction methods must also be suitable for dealing with uncertainty and for ongoing effort estimation.

At the same time, in this process, the relationship between project effort and the subset of features that affect the effort is unclear or the relational information is incomplete[1]. On the other hand, software project data sets are usually small. It can be difficult to select feature subsets and predict project effort with traditional statistical methods or machine learning methods. These methods usually require a large complete sample or data that follow a certain statistical distribution.

Unfortunately, current effort prediction methods do not properly take into account these critical characteristics of the software development process and the project data sets. Although there is a great deal of research activity on this topic, a wide range of prediction techniques being proposed, and increasing numbers of empirical studies published, no one technique is consistently effective. The software industry does not have a good track record for accurate cost estimation with typically 75% of projects reporting overruns[2]. Software project effort prediction is a form of systems engineering that demands systematic solutions. Specifically, we should systematically address effort prediction in the context of small, incomplete, and noisy data sets. This includes outlier detection, missing data imputation, feature subset selection, and continuous effort prediction.

Grey System Theory (GST), a recently developed system engineering theory based on the uncertainty of small samples, was first developed by Deng in 1982 [11]. The system was named by using grey as the color which indicates the amount of known information in control theory. For instance, if the internal structures and features of a system are completely unknown, the system is usually denoted as a 'black box'. In contrast, 'white' means that the internal features of a system are fully explored. Between white and black, there exist grey systems indicating that some of the information is clear, whilst other aspects are still unknown.

One distinct advantage of GST is that it allows us to utilize only a few known data to estimate the behavior of an uncertain system. In the context of data-starvation, GST is known to be effective and has been widely and successfully applied to address the real world problems in image processing [23], mobile communication [47], machine vision inspection [21], decision making [32], stock price prediction [50], and system control [17]. These successful applications inspire us to explore the GST-based approach to systematically address project effort prediction questions.

Grey Relational Analysis(GRA) is a method of GST. It is a distinct similarity measurement that is different from traditionally distance measure [31]. Further, some studies [45, 15, 16, 18] have shown that the GRA method performs acceptably in predicting software development effort. However, we found that these studies lay emphases on the different aspects of software project effort prediction and none of them consider noisy data problem.

Unfortunately, Almost all the real world software project data sets contain outliers which are the data objects that do not comply with the general behavior or model of the data. Specifically, in the context of software project effort prediction, an outlier is a project which with the comparable feature values but incomparable effort with those of other projects. The outliers can be problematic for accurate prediction or pattern identification. Thus, we firstly address this problem in the context of software effort prediction.

On the other hand, the features are designated to characterize a set of software projects, some of them can

[1]Not all features (variables) that are available are useful and some may not only be redundant, they may also positively hinder the prediction task. Removing such features is known as feature subset selection. For more details see Kohavi and John [29]

[2]See for example Moløkken and Jørgensen [36].

be used to estimate software project effort. Feature subset selection, which is the process of identifying and removing as many irrelevant and redundant features as possible from an original feature set for the purpose of providing better prediction accuracy, can help us obtain useful features from software project data. It has been widely used to obtain predictive features without creating new features based on transformations or combinations of the original feature set. However, most of the feature subset selection methods are for classification tasks and tend to give poor results when the sample size is small [19]. At the same time, software project effort is a continuous valued feature that makes classification methods generally inappropriate. Therefore, we also seek to improve upon traditional feature subset selection methods.

In this paper, we intend to systematically address the data quality, feature subset selection and weighted determinations for both features and efforts problems for the purpose of software effort prediction with between-project data sets [3] at an early stage of a development process.

Our experiments, using five publicly available real-world data sets, explore the advantages of our GRACE$^+$ method, which is an enhancement of GRACE [45], might offer relative to existing methods.

The remainder of the paper is organized as follows. In the next section we present related work on software effort prediction. This is followed by the introduction of the Grey Relational Analysis (GRA) method. After that, we describe the proposed GRA based effort prediction method GRACE$^+$. The experiments and results follow with a concluding discussion and suggestions for further work.

2 Related Work

Machine learning methods have been used to predict software effort since the 1990s with advantages that they are nonparametric and adaptable, and many of them make no or minimal assumptions about the form of the function under study [46]. Of course, there are also problems, not least that they are difficult to configure to new situations and there is little theory so that their application often becomes a matter of trial and error. The most commonly used methods include case based reasoning (CBR), regression and decision trees, and artificial neural networks (ANN).

Mukhopadhyay et al. [37] presented early work using CBR. They used Kemerer's data set [26] and found that their analogy-based model Estor outperformed COCOMO [2]. Shepperd and Schofield [42] compared an analogy based method with stepwise regression (SWR) on nine different industrial data sets and report that in all cases analogy equaled or outperformed SWR. Finnie and Wittig [14] compared CBR with different regression models using function points (FPs) and ANNs. They report that CBR outperformed regression models based on function points. On the other hand, Briand et al. [7] and Jørgensen et al. [22] obtained conflicting results where the regression model generated significantly better results than a CBR approach. Finnie and Wittig [14] also found ANN outperformed CBR.

Using regression and decision trees is another method of software effort prediction. Briand et al. [6] compared the optimized set reduction (OSR) strategy with COCOMO and SWR. They used the COCOMO81 data set [3] as a training set and the Kemerer data set as a test set. OSR outperformed COCOMO and

[3]The data objects of a between-project data set come from different projects

SWR. Srinivasan and Fisher [46] illustrated the use of CARTX to predict software effort. They also used the COCOMO81 data set for training and the Kemerer data set for testing. They report that CARTX outperformed COCOMO and SLIM [39].

Wittig and Finnie [52] report the use of back propagation (BP) neural networks to predict software effort. Although the results are encouraging, unfortunately though the data sets were large only a small number of projects were used for testing. Srinivasan and Fisher [46] found BP neural networks outperformed regression trees. Samson et al. [41] compared an ANN method with linear regression using the COCOMO81 data set. Although the ANN method outperformed the linear regression method, both methods performed badly.

For a more detailed review see Briand and Wieczorek [8]. Nonetheless, even this brief introduction to machine learning methods for software effort prediction reveals that the results are quite mixed. Apart from using different data sets and different experimental methods, these methods tend to borrow from other disciplines where large data sets are the norm or data that follow a certain statistical distribution are necessitated.

Recently, GRA was used to predict software effort. Song et al. [45] firstly introduced a GRA method called GRACE to predict software development effort. Hsu et al. [15, 16] proposed an improved GRA model to enhance predicted results. Huang et al. [18] integrated a genetic algorithm (GA) to the GRA. The GA method is adopted to find the best fit of weights for each software effort driver in the similarity measures. Wang et al. [49] employed Grey model to predict software stage-effort. However, outliers detection, feature subset subsection, and weighted determinations for both features and efforts were not completely considered. Our GRACE$^+$ method, which based on GRACE [45], aimed to systematically address these issues, it is quite different from these methods.

3 Grey Relational Analysis

Many statistical correlation analyses can provide not only correlation coefficients among factors but also the relevant significance level. In most cases, data distribution is assumed as linear, exponential or logarithmic, and errors are normally distributed with zero means. Moreover, sufficient data are required to determine its distribution type and to ensure statistical significance. However, it may difficult to get adequate information in many situations such as those relating to the software effort prediction issues. Under such conditions, precise analytical results may not be available. Grey relational analysis (GRA) provides an alternative approach to identify the correlations among factors, or to build prediction models in the context of data-starvation.

GRA works in the grey relational space, which was proposed by Deng [11] based on the combined concepts of system theory, space theory and control theory. It can be used to capture the correlations between the reference factor and other compared factors of a system [12]. One of the features of GRA is that both qualitative and quantitative relationships can be identified among complex factors with insufficient information (relative to conventional statistical methods). Under such a condition, the results generated by conventional statistical techniques may not be acceptable without sufficient data to achieve desired confidence levels. In contrast, GRA can be used to identify major correlations among factors of a system with a relatively small amount of data. Thus, one of the major advantages of GRA is that it can generate satisfactory outcomes using a relatively small amount of data or with great variability in factors since it can increase the data regularity with proper data treatment [30].

GRA, whose mathematics is derived from space theory [12], is used to quantify the influence of a compared series on the reference series. The degree of influence, which is referred to as the grey relational grade, can be represented by the relative distance between them in an imaging grey space without making prior assumption about the distribution type. The smaller the distance, the larger the influence.

Generally, the procedure of GRA consists of the following two steps.

Step 1: The generation of grey relation. In this step, GRA removes anomalies associated with different measurement units and scales by the normalization of raw data which usually is called the generation of grey relation.

Step 2: The calculation of grey relational grade. GRA uses the grey relational coefficient to describe the trend relationship between an objective series and a reference series at a given (time) point in a system.

Let $X = \{x_\sigma | \sigma = 0, 1, 2, \ldots, n\}$ be a given grey relational factor set, suppose $x_i = \{x_i(1), x_i(2), \ldots, x_i(m)\}$ is a data series, where $x_i(k) \in X$ is the value of x_i $(i \in \{0, 1, 2, \ldots, n \in N\})$ at (time) point $k(1 \leq k \leq m \in N)$. Suppose x_0 is the reference series and $x_1, x_2, \ldots, x_n$ are the objective series, the grey relational coefficient $\gamma(x_0(k), x_i(k))$ between the reference series x_0 and the objective series $x_i(i \in \{1, 2, \ldots, n\})$ at (time) point $k \in \{1, 2, \ldots, m\}$ was defined by Deng as follows:

$$\gamma(x_0(k), x_i(k)) = \frac{\Delta_{min} + \zeta\Delta_{max}}{\Delta_{0,i}(k) + \zeta\Delta_{max}}, \tag{1}$$

where

1. $$\Delta_{0,i}(k) = \begin{cases} |x_0(k) - x_i(k)| & \text{if } x_0(k) \text{ and } x_i(k) \text{ are numericals} \\ 1 & \text{if } x_0(k) \text{ and } x_i(k) \text{ are categoricals and } x_0(k) \neq x_i(k); \\ 0 & \text{if } x_0(k) \text{ and } x_i(k) \text{ are categoricals and } x_0(k) = x_i(k) \end{cases}$$

2. $\Delta_{min} = min_j min_k \Delta_{0,j}(k)$ is the smallest value of $\Delta_{0,j}(k)$, $\forall j \in \{1, 2, \ldots, n\} \wedge \forall k \in \{1, 2, \ldots, m\}$;

3. $\Delta_{max} = max_j max_k \Delta_{0,j}(k)$ is the largest value of $\Delta_{0,j}(k)$, $\forall j \in \{1, 2, \ldots, n\} \wedge \forall k \in \{1, 2, \ldots, m\}$;

4. $\zeta \in (0, 1]$ is a distinguishing coefficient used to adjust the range of the comparison environment, and to control the level of differences of the relational coefficients. When $\zeta = 1$, the comparison environment is unaltered; when $\zeta = 0$, the comparison environment disappears. In cases where data variation is large, ζ usually changes from 0.1 to 0.5 for reducing the influence of extremely large Δ_{max}. Although the ζ value will change the absolute value of $\gamma(x_0(k), x_i(k))$ but will not change the ranking order of $\gamma(x_0(k), x_i(k))$ among the n series.

As there are too many relational coefficients to be compared directly, further data reduction makes use of average-value processing to convert each series' grey relational coefficients at all (time) points into its mean. The mean is also referred to as the grey relational grade.

The grey relational grade $\Gamma(x_0, x_i)$ between an objective series $x_i (i \in \{1, 2, \ldots, n\})$ and the reference series x_0 was defined by Deng as follows:

$$\Gamma(x_0, x_i) = \frac{1}{m} \sum_{k=1}^{m} \gamma(x_0(k), x_i(k)). \tag{2}$$

However, in practice the influence of each factor on a system is not exactly the same, for example, Eqn. 2 can be modified as follows:

$$\Gamma(x_0, x_i) = \sum_{k=1}^{m} \beta_k \gamma(x_0(k), x_i(k)), \tag{3}$$

where β_k is the normalized weight for factor $k \in \{1, 2, \ldots, m\}$, and $\sum_{k=1}^{m} \beta_k = 1$. It can be set to any value that reflects the relative importance of all the series $x_i (i \in \{1, 2, \ldots, n\})$ to series x_0 at (time) point k.

It should be noted that:

1. The aim of GRA is to recognize the geometric relationship between two sets of (time) series data in relational space. If the data of the two series at all respective (time) points are the same, then the relational coefficients all equal one, as does the relational grade. On the other hand, since it is impossible for any two transformed series to be perpendicular to one another, the grey relational coefficients are greater than zero, as is the relational grade.

2. The grey relational grade between two factors is non-unique. Generally speaking, one of the main purposes of relation analysis is to find the importance order of factors, so the rankings of relational grades are more significant [51]. Yet, the relational order may change slightly with different ζ and different original reference points. If the ranking of the relational grade of a factor remains unchanged for any condition, we call it a factor with 'white' relational grade, and the relevant results are more reliable. Presently, there is no criterion having been set in most grey relation analyses regarding sensitivity of ζ values to justify significance of relational grade. In this study, we learn the value of ζ from the given data sets, please see subsection 4.3 for details.

4 GRA Based Software Effort Prediction Method

Software project effort prediction is system engineering that demands us systematically to address the data quality and feature subset selection problems before predicting software effort. Thus the GRA based software effort prediction method GRACE$^+$ consists of the following three components(See Fig. 1):

1. GRA based outlier detection (OD@GRACE$^+$). In this optional component, by identifying and removing outliers contained in a software project data set, we prepare quality data for feature subset selection and further for software effort prediction.

2. GRA based feature subset selection (FSS@GRACE$^+$). In this component we use the proposed feature subset selection method to choose the predictive features for software effort prediction purpose.

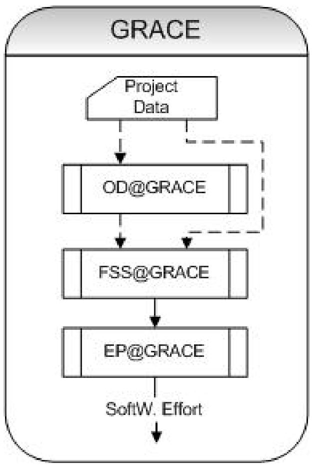

Figure 1: The structure of GRACE$^+$

3. GRA based effort prediction (EP@GRACE$^+$). In this component we estimate software effort using the proposed method with the data that have been selected by the proposed outlier detection and feature subset selection methods.

4.1 Detecting Outliers with GRA

As all machine learning methods induce knowledge strictly from data, the quality of the knowledge extracted is largely determined by the quality of the underlying data. This makes any single machine learning method potentially suffer from garbage in garbage out problems. Therefore, data quality has been one of the major concerns in the machine learning field, and quality data is a precondition for obtaining quality knowledge. Unfortunately, almost all the real world data sets are companied by outliers. This is one factor that affects data quality. Outlier detection can be helpful for improving data quality. We propose a GRA based outlier detection method in the context of software effort prediction.

A software project data set is characterized by a set of features which also can be used to predict software project effort. That is, *Effort* $= \mathcal{F}($*Features*$)$. From this we can deduce that if the feature values of a project are comparable to those of other projects, then the corresponding effort should be comparable to those of other projects as well. Otherwise, the project can be an outlier in the context of software project effort prediction. That is, an outlier is a project which with the comparable feature values but incomparable effort with those of other projects. Thus, when searching for an outlier, the feature of software effort is taken into account if and only if other features' values are comparable. This makes our method quite different from the other distance-based or those density-based general purpose outlier detection methods [28, 5] .

Suppose $\mathcal{D} = \{p_1, p_2, \dots, p_i, \dots, p_n\}$ is a software project data set, and $p_i = \{f_i(1), f_i(2), \dots,\ f_i(m), e_i\}$ where $i = \{1, 2,\ \dots, n \in N\}$ is a project. For each project $p_i \in \mathcal{D}$, $f_i(1), f_i(2), \dots,$ and $f_i(m)$ are the feature data of the project, and e_i is the corresponding effort. Then $\mathcal{D}$ can be represented in the form of a matrix as:

$$\mathcal{D} = \begin{pmatrix} f_1(1) & f_1(2) & \dots & f_1(m) & e_1 \\ f_2(1) & f_2(2) & \dots & f_2(m) & e_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ f_i(1) & f_i(2) & \dots & f_i(m) & e_i \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ f_n(1) & f_n(2) & \dots & f_n(m) & e_n \end{pmatrix}. \tag{4}$$

When predicting software project effort with the GRA based method, we use the most influential projects, which can be denoted as $\mathcal{I} \subset \mathcal{D}$, to estimate the effort of a project $p_i \in \mathcal{D}(i \in \{1, 2, \dots, n\})$. Therefore, according to the definition of an outlier described above, if the feature values of a project $p_j \in \mathcal{I}(j \in \{1, 2, \dots, n\} \wedge j \neq i)$ are comparable to those of p_i but its effort is highly distinct to that of the latter, then p_j can be an outlier for the purpose of predicting the effort of p_i. Suppose for $\forall p_i \in \mathcal{D}$, a set $\mathcal{O} = \cup p_j$ is the candidate outlier set in which false outliers must be detected. If $o_1 \in \mathcal{O}$ is one of the most influential projects for $o_2 \in \mathcal{O}$ and the mutual distance [4] between them is small enough, then o_1 and o_2 are false outliers and should be removed from the candidate outlier set $\mathcal{O}$. Thus $\mathcal{O} \ominus \cup o_f$ [5] is the outlier set, where o_f is a false outlier. The mutual distance is small enough means it is smaller than a given threshold. The threshold is set to 2*k, where k is the number of the most influential projects. The value of k is learned from the given data set, please refer to Section 4.3 for details.

To summarize, the GRA based outlier detection method consists of two steps: the candidate outlier discovery and the false outlier filter. Viewing each of the n projects as a data series, the outlier detection procedure can be described as follows:

1. *The candidate outlier discovery.* For a project $p_i \in \mathcal{D}(i \in \{1, 2, \dots, n\})$, first we calculate the grey relational grades $\Gamma(p_i \ominus \{e_i\}, p_j \ominus \{e_j\})$ between projects p_i and each of $p_j(j \in \{1, 2, \dots, n\} \wedge j \neq i)$; then by sorting the $n - 1$ projects according to the strategy of the projects in which those with larger grey relational grades have higher priority, we obtain the most influential projects $\mathcal{I}$ which consists of the first several projects; finally, viewing each e_j of project $p_j \in \mathcal{I}$ as the prediction of e_i of project p_i, we calculate the relative error ε of e_i. If ε is greater than a given threshold θ_e, then the corresponding project p_j is a candidate outlier. For the relative error threshold, we let it be 0.8, but it is adjustable. Repeat this process until all projects are covered, we obtain a candidate outlier set $\mathcal{O}$.

2. *The false outlier filter and final outlier set formation.* For each outlier $o_1 \in \mathcal{O}$, we respectively calculate the mutual distance d_{o_1,o_2} between it and each of the other outliers $o_2 \in \mathcal{O} \ominus \{o_1\}$, the relative errors ε_1 of the project corresponding to o_1 and ε_2 of the project corresponding to o_2. If the mutual distance

[4]The mutual distance between two points is defined as the sum of the rankings of the points respectively in the ranking list of another one. For example, suppose there are four points A, B, C, and D. The distance between A and B is 3.23, between A and C is 2.15, between A and D is 2.1; the distance between B and C is 1.56, between B and D is 4.32. For point A, the ranking list is {D, C, B} and the ranking of B in A's list is 3; for point B, the ranking list is {C, A, D} and the ranking of A in B's list is 2. Thus the mutual between A and B is 3 + 2 = 5. See [44] for details.

[5]The notation $\mathcal{O} \ominus \cup o_f$ means set $\mathcal{O}$ minus set $\cup o_f$.

```
Input: $\mathcal{D}$ – the software project data set,
       $\theta_d$ – the mutual distance threshold,
       $\theta_e$ – the relative error threshold.
Output: $\mathcal{O}$ – the outliers in the data set $\mathcal{D}$.
1: $\mathcal{O} \leftarrow \emptyset$, $\mathcal{O}_f \leftarrow \emptyset$; //$\mathcal{O}$ and $\mathcal{O}_f$ are an outlier set and a false outlier set respectively
2: for each project $p_i \in \mathcal{D}$ do //the candidate outlier discovery
3:     compute $\Gamma(p_i \ominus \{e_i\}, p_j \ominus \{e_j\})$ for $\forall j \in \{1, 2, \ldots, n\} \wedge j \neq i$;
4:     sort the $n-1$ projects according to the grey relational grades, the largest first;
5:     $\mathcal{I} \leftarrow$ { the first $\theta_d/2$ projects };//the influential projects for $p_i$
6:     for each project $p_j \in \mathcal{I}$ do
7:         compute the relative error $\varepsilon$ of project $p_i$;
8:         if $\varepsilon \geq \theta_e$ then $\mathcal{O} \leftarrow \mathcal{O} \cup p_j$;
9:     end for
10: end for
11: for each outlier $o_1 \in \mathcal{O}$ do //the false outlier filter
12:     for each outlier $o_2 \in \mathcal{O} \ominus \{o_1\}$ do
13:         compute the mutual distance $d_{o_1,o_2}$;
14:         compute the relative errors $\varepsilon_1$ of $o_1$ and $\varepsilon_2$ of $o_2$;
15:         if $d_{o_1,o_2} \geq \theta_d \wedge \varepsilon_1 \geq \theta_e \wedge \varepsilon_2 \geq \theta_e$ then $\mathcal{O}_f \leftarrow \mathcal{O}_f \cup \{o_1, o_2\}$;
16:     end for
17: end for
18: $\mathcal{O} \leftarrow \mathcal{O} \ominus \mathcal{O}_f$;
```

Table 1: The algorithmic description of the OD@GRACE$^+$ method

d_{o_1,o_2} is greater than a given threshold θ_d and the relative errors ε_1 and ε_2 are greater than a given threshold θ_e, then o_1 and o_2 are false outliers. Up to now, we otain a temporary false outlier set $\mathcal{O}_f = \mathcal{O}_f \cup \{o_1, o_2\}$ and let $\mathcal{O} = \mathcal{O} \ominus \mathcal{O}_f$. Repeat this process for all the rest outliers $\mathcal{O}$, finally we obtain the false outlier set $\mathcal{O}_f$. Then $\mathcal{O} \ominus \mathcal{O}_f$ is the outlier set.

The algorithmic description of the method is shown in Table 1.

4.2 Selecting Feature Subset with GRA

The features are designated to characterize a set of software projects, some of them can be used to estimate software project effort. Therefore, before predicting software effort, we should firstly decide which features are useful for the specific prediction task. This is referred to as feature subset selection (FSS) [9, 27, 43, 19], which is the process of identifying and removing as much irrelevant and redundant information as possible from an original feature set for the purpose of providing better prediction accuracy.

The main FSSs can be divided into two categories: the wrapper and filter methods [29]. A wrapper method uses a predetermined selection algorithm to search for feature subsets, and employs a induction algorithm to evaluate them iteratively and make the final decision. The same induction algorithm will be used to induce the final target concept. These types of methods can obtain high induction accuracy but inherit the limitations of the induction algorithm and are highly expensive in terms of the computational cost. A filter method does not take into account the biases of the induction algorithms and selects feature subsets that are independent of the induction algorithms. As measures can be devised that are algorithm specific, they may be computed efficiently and hence can be applied to large data sets with many features. However, there is a danger that features selected by these types of methods cannot allow an induction algorithm to reach its maximum accuracy.

Unfortunately, both the wrapper and the filter methods have mainly been explored for classification tasks. At the same time, for the conventional feature subset selection algorithms, the wrapper and the filter methods tend to give poor results when the sample size is small [19]. This means they can struggle to select a feature subset for predicting software project effort which is a continuous valued feature, and a software project data set usually is small. Therefore, we propose a feature subset selection method FSS@GRACE$^+$ based on GRA, which was proposed for providing reliable results with small amount of data. Meanwhile, FSS@GRACE$^+$ is designed as a wrapper method, so a high prediction accuracy could be obtained. As the wrapper methods are expensive in terms of the computational cost, so if we use this type of method to choose feature subset, more time will be consumed. Fortunately, the smallness of a software project data set minimizes the computational cost and makes it acceptable.

The FSS@GRACE$^+$ method consists of two steps: the importance evaluation and the optimal feature subset search. The former step evaluates the importance of all other features to software effort, and sorts them according to the corresponding grey relational grades, the largest first. The result is a priority list $\mathcal{L}_f$. In the latter step, the forward selection strategy is used to select the optimal feature subset $\mathcal{S}$ from $\mathcal{L}_f$. That is, it begins with a feature subset $\mathcal{S}$ with only one feature which is the first one in $\mathcal{L}_f$, and computes the relative error ε_l after invoking EP@GRACE$^+$ (see Subsection 4.3 for details); then expands $\mathcal{S}$ with the next feature in $\mathcal{L}_f$ and computes the relative error ε_c; terminates if $\varepsilon_c > \varepsilon_l$, otherwise $\varepsilon_l \leftarrow \varepsilon_c$, expands $\mathcal{S}$ and continues the process (see Table 2 for details).

As the optimal feature subset search step is straightforward and has been introduced, we focus only on the first step hereafter.

From the properties of Γ (see Section 3 for details) we know that if an input series shows a higher influence on the output than others, then the series can be considered to be more important to the output. We view the feature data as input series and software effort as output and apply GRA to evaluate the feature importances, the specific procedure is as follows:

1. *Data series construction.* Viewing each of the column vectors of matrix $\mathcal{D}$ as a data series, we obtain a total of $m+1$ series. These series are: $f(1) = \{f_1(1), f_2(1), \ldots, f_n(1)\}$, $f(2) = \{f_1(2), f_2(2), \ldots, f_n(2)\}$, $\ldots$, $f(m) = \{f_1(m), f_2(m), \ldots, f_n(m)\}$, and $f(0) = \{e_1, e_2, \ldots, e_n\}$.

2. *Grey relational grade calculation.* Viewing $f(0)$ as the reference series and $f(1), f(2), \ldots, f(m)$ as

```
Input: $\mathcal{D}$ – the software project data set.
Output: $\mathcal{S}$ – the selected feature subset.
  // the importance evaluation
1: for each feature $f(i) \in \mathcal{D}(i \in \{1, 2, \ldots, m\})$ do
2:     compute the grey relational grade $\Gamma(f(0), f(i))$;
3: end for
4: sort the $m$ features according to the grey relational grades, the largest first;
5: $\mathcal{L}_f \leftarrow$ { the sorted $m$ features };
  // the optimal feature subset search
6: $\mathcal{S} \leftarrow \mathcal{L}_f(1)$, $\varepsilon_l \leftarrow 0$, $\varepsilon_c \leftarrow 0$;
7: for each project $p_i \in \mathcal{D}(i \in \{1, 2, \ldots, n\})$ do
8:     invoke EP@GRACE$^+$ with $\mathcal{S}$, and compute the relative error $\varepsilon_i$ for $p_i$;
9:     $\varepsilon_l \leftarrow \varepsilon_l + \varepsilon_i$;
10: end for
11: expand $\mathcal{S}$ with the next feature in $\mathcal{L}_f$;
12: for each project $p_i \in \mathcal{D}(i \in \{1, 2, \ldots, n\})$ do
13:     invoke EP@GRACE$^+$ with $\mathcal{S}$, and compute the relative error $\varepsilon_i$ for $p_i$;
14:     $\varepsilon_c \leftarrow \varepsilon_c + \varepsilon_i$;
15: end for
16: if $\varepsilon_c > \varepsilon_l$ then
17:     return $\mathcal{S} \ominus$ { the last feature in $\mathcal{S}$};
18: else
19:     $\varepsilon_l \leftarrow \varepsilon_c$, $\varepsilon_l \leftarrow 0$, goto 11;
20: end if
```

Table 2: The algorithmic description of the FSS@GRACE$^+$ method

the corresponding objective series, respectively compute the grey relational grade $\Gamma(f(0), f(i))$ ($i \in \{1, 2, \ldots, m\}$) with Eqn. 2.

3. *Feature sort.* Using the Quick-sort algorithm to sort the features according to the corresponding grey relational grades $\Gamma(f(0), f(i))(i \in \{1, 2, \ldots, m\})$, the largest first. At the same time, $wof_i = \Gamma(f(0), f(i))$ will be normalized and used as the weight of feature $f(i)$ by EP@GRACE$^+$. The result is a priority list $\mathcal{L}_f$ which will be submitted to the optimal feature subset search step.

The algorithmic description of the method is shown in Table 2.

4.3 Predicting Software Effort with GRA

GRA was originally used to measure the importance of objective series to a reference series. We view project data as series, specifically a new project is supposed to be a reference series and other projects are regarded as objective series, and apply this technique here to select projects that exhibit a stronger impact on the new project i.e. the one for which we wish to predict effort. Then we use the selected projects' effort to estimate that of the new project. Viewing each of the n projects as a data series, the specific effort prediction procedure is as follows (see Table 3 for the algorithmic description):

1. *Grey relational grade calculation.* Viewing p_1 as the reference series and $p_2, p_3, \ldots, p_n$ as the corresponding objective series, respectively compute the grey relational grade $\Gamma(p_1 \ominus \{e_1\}, p_i \ominus \{e_i\})(i \in \{2, 3, \ldots, n\})$ with Eqn. 3, where $\beta_i = wof_i$ which is provided by FSS@GRACE$^+$ (please see subsection 4.2 for details).

2. *Project sort.* Using the Quick-sort algorithm to sort the projects according to the corresponding grey relational grades $\Gamma(p_1 \ominus \{e_1\}, p_i \ominus \{e_i\})(i \in \{2, 3, \ldots, n\})$, the largest first. The result is a priority list $\mathcal{L}_p$, some of them will be used to predict the effort of the new project p_1.

3. *Effort prediction.* Aggregate the effort of the first k projects in $\mathcal{L}_p$ with the corresponding weight as the prediction of project p_1. Specifically, it is defined as follows:

$$\hat{\mathcal{E}}(p_1) = \sum_{i=1}^{k} w_i \times \mathcal{E}(p_i), \tag{5}$$

   where $\mathcal{E}(p_i)$ is the effort of the ith most influential project p_i, $w_i = \frac{\Gamma(p_1 \ominus \{e_1\}, p_i \ominus \{e_i\})}{\sum_{j=1}^{k} \Gamma(p_1 \ominus \{e_1\}, p_j \ominus \{e_j\})}$ is the normalized contribution of project p_i to project p_1, and $\Gamma(p_1 \ominus \{e_1\}, p_h \ominus \{e_h\})(h \in \{i, j\})$ is the grey relational grade between p_1 and p_h.

However, in this procedure, there is an outstanding problem, i.e. the selection of the grey relational grade computation method g, the grey distinguishing coefficient ζ, and the number of the most influential projects k. Since our purpose is predicting software effort from small data sets, we preset $\mathcal{K} = \{1, 2, 3, 4, 5\}$. For each method g and each $k \in \mathcal{K}$, we learn ζ from the objective series – the projects excluding the new project. Specifically, by changing ζ from 0 exclusive to 1 with a predefined increment of δ, we predict effort for each of the projects in the objective series using method g with the pair of (k, ζ). By analyzing the relationship between the relative errors and these two parameters, we obtain the optimal g, k and ζ which are finally used to predict software effort.

The algorithmic description of the method is shown in Table 4.

5 Experiments and Results

5.1 Data sources

Different researchers usually use different data sets to test their methods which makes it hard to compare their results. In order to compare our results with other researchers' results, and allow other researchers to confirm our results, we used five publicly available data sets as our data source. Most of these data sets have been used by more than one researcher to evaluate software cost estimation models. For example,

```
Input: $\mathcal{D}$ – the software project data set,
           $k$ – the number of the most influential projects,
           $\zeta$ – the grey distinguishing coefficient,
           $g$ – the grey relational grade computation method.
Output: $\mathcal{E}$ – the effort of a new project.
Function EP($\mathcal{D}$, $k$, $\zeta$, $g$, $\mathcal{E}$)
1: for each project $p_i \in \mathcal{D}(i \in \{2, 3, \ldots, n\})$ do
2:     if $g$ is not Wu's method then
3:         compute the grey relational grade $\Gamma(p_1 \ominus \{e_1\}, p_i \ominus \{e_i\})$ using the $g$ method with $\zeta$ ;
4:     else
5:         compute the grey relational grade $\Gamma(p_1 \ominus \{e_1\}, p_i \ominus \{e_i\})$ ;
6:     end if
7: end for
8: sort the $n$-1 projects according to the grey relational grades, the largest first;
9: $\mathcal{L}_p \leftarrow$ { the first $k$ projects };
10: return $\mathcal{E} \leftarrow$ { the result of Eqn. 5 };
```

Table 3: The algorithmic description of the Function EP of the EP@GRACE$^+$ method

COCOMO81 is the data set that was used by Boehm [3] to build the COCOMO model and also was used by Briand et al. [6], Srinivasan and Fisher [46], and Samson et al. [41] to compare different effort prediction methods; the Kemerer data set was coded by Kemerer [26] using the same attributes as Boehm and later was used by Briand et al. [6], Srinivasan and Fisher [46], and Shepperd and Schofield [42]; the Albrecht data set actually is the IBM DP Services data but was first used by Albrecht and Gaffney [1] and was also used by Shepperd and Schofield [42] to validate software size and effort estimation methods.

Tables 5 and 6 summarize these data sets from different points of view. These tables show that application domains range from business, science, and technology to system software. The project size ranges from 1.98 KSLOC to 1150 KSLOC (or from 62 FPs to 1116 FPs), the project effort ranges from 0.5 person months to 11400 person months (or from 546 person hours to 23940 person hours), and the number of projects ranges from 15 to 77. Therefore, the data sets are quite diverse.

To summarize, in order to avoid the results are obtained is due to any kind of characteristic or peculiarity of data sets, 1) the data sets we used are quite diverse in terms of project size, project effort, and the number of projects; 2) the data sets are publicly available and most of them have been used by more than one researcher to evaluate software cost estimation models.

```
Input: $\mathcal{D}$ – the software project data set,
          $\delta$ – the increment of grey distinguishing coefficient.
Output: $\mathcal{E}$ – the effort of a new project.
   // learning $k$ and $\zeta$ from historical data
1: $\mathcal{L}_e \leftarrow \emptyset$; // a 3-dimension array which is used to store the relative errors
2: for each method $g$ do
3:    for each $k \in \mathcal{K}$ do
4:       if $g$ is Wu's method then go to 6;
5:       for each $\zeta \in (0, 1]$ with an increment of $\delta$ do
6:          for each historical project $p_i \in \mathcal{D}$ do
7:             invoke EP($\mathcal{D}$, $k$, $\zeta$, $g$, $\mathcal{E}_0$);
8:             compute the relative error $e_0$ of $e_i$ according to $\mathcal{E}_0$;
9:             $e \leftarrow e + e_0$ ;
10:         end for
11:         $\mathcal{L}_e \leftarrow \mathcal{L}_e \cup \{e\}$ ;
12:         if $g$ is Wu's method then go to 14;
13:     end for
14:  end for
15: end for
16: search for the optimal $g_0$, $k_0$, and $\zeta_0$ from $\mathcal{L}_e$ according to $e$;
17: invoke EP($\mathcal{D}$, $k_0$, $\zeta_0$, $g_0$, $\mathcal{E}_0$);
18: $\mathcal{E} \leftarrow \mathcal{E}_0$;
```

Table 4: The algorithmic description of the EP@GRACE$^+$ method

Name	Projects	Features	Description	Source
Albrecht	19	8 (Language, IN, OUT, FILE, INQ, FP, WorkHours)	IBM DP Services projects	[1]
COCOMONASA	60	17 (rely, data, cplx, time, stor, virt, turn, acap, aexp, pcap, vexp, lexp, modp, tool, sced, ksloc, actual effort)	Aerospace applications from 1980s to 1990s	[34]
COCOMO81	63	17 (rely, data, cplx, time, stor, virt, turn, acap, aexp, pcap, vexp, lexp, modp, tool, sced, ksloc, actual effort)	Business, scientific, and system software projects	[3]
Desharnais	77	8 (Team Exp., Manager Exp., Length, Transactions, Entities, Points non ajust., Language, Effort)	Canadian software house, commercial projects	[13]
Kemerer	15	17 (rely, data, cplx, time, stor, virt, turn, acap, aexp, pcap, vexp, lexp, modp, tool, sced, ksloc, actual effort)	Large business applications	[26]

Table 5: Data sets used in the experiments

Dataset	Size (KSLOC/FP)	Effort (PM/PH)
Albrecht	Mean = 61.08, Min = 3, Max = 318	Mean = 21.88, Min = 0.5, Max = 105.2
COCOMONASA	Mean = 74.59 Min = 2.2, Max = 423	Mean = 406.41, Min = 8.4, Max = 3240
COCOMO81	Mean = 77.21, Min = 1.98, Max = 1150	Mean = 683.32, Min = 5.9, Max = 11400
Desharnais	Mean = 282.39 FPs, Min = 62 FPs, Max = 1116 FPs	Mean = 4833.91 PHs, Min = 546 PHs, Max = 23940 PHs
Kemerer	Mean = 184.37, Min = 39, Max = 450	Mean = 219.25, Min = 23.2, Max = 1107.31

Table 6: Descriptive statistics for project size and effort

5.2 Experimental method

5.2.1 Validation method

In practice, software project data sets are frequently small. The holdout method makes inefficient use of the data, because generally a third of the data set is hidden from the prediction method. In order to more efficiently use data and cover all projects, we used the *jack knife* methodology to validate the proposed GRACE$^+$ method. Specifically, for each of the n projects of a given data set, we predicted its effort with (k, ζ) which were learned from the remaining $n-1$ projects. The evaluation measures defined in subsection 5.2.2 are then computed over the n predictions.

For the purpose of evaluating the performance of the proposed GRACE$^+$ method, we compared it with both the stepwise regression method (SWR) and the Analogy method [42]. For SWR, the prediction models were generated using the entire data set. This means the results are likely to be slightly biased in favor of the regression models. For Analogy, we let $k = 2$ [6], and an exhaustive feature subset search method was used. This means Analogy can obtain its optimum prediction accuracy.

In addition, we also compared our results with NW GRA [16], LW GRA [16], ANN [41, 18], CART [46, 18], GRA [18], and GP [10] methods where the corresponding results and evaluation procedure are comparable. Here, the comparable means that the data sets, the validation method and the evaluation criteria are the same.

5.2.2 Evaluation criteria

A common criterion for evaluating software effort prediction methods is the Magnitude of Relative Error (*MRE*). For a project i whose effort is predicted, the corresponding MRE_i is defined as follows:

$$MRE_i = \frac{|\mathcal{E}_i - \hat{\mathcal{E}}_i|}{\mathcal{E}_i} \times 100, \tag{6}$$

where $\hat{\mathcal{E}}_i$ is the prediction of effort $\mathcal{E}_i$.

By averaging MRE_i over multiple projects n, *M*ean *MRE* (*MMRE*) is obtained:

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i. \tag{7}$$

As *MMRE* is sensitive to individual predictions with excessively large *MRE*s, we also use the median of *MRE*s for the n projects (*MdMRE*), which is less sensitive to extreme values, as another measure.

For both *MMRE* and *MdMRE*, a higher score means worse prediction accuracy.

When using *MRE* as a measure of prediction accuracy, we suppose the error is proportional to the size of the project. Thus, *PRED(l)* is usually used as a complementary criterion. This is defined as follows:

$$PRED(l) = \frac{k}{n} \times 100 \tag{8}$$

[6] We used the two analogies (k=2) since Kadoda et al. [24] suggested this to perform consistently better than higher values of k for this particular problem domain.

	Reduced Number of	
Data Set	Features (%)	Projects (%)
Albrecht	5 (62.5%)	1 (5.3%)
COCOMONASA	8 (47.1%)	2 (3.3%)
COCOMO81	3 (17.7%)	1 (1.6%)
Desharnais	2 (25.0%)	2 (2.6%)
Kemerer	8 (47.1%)	1 (6.7%)

Table 7: The reduced numbers of features and projects for the five data sets

where k is the number of projects in which $MRE_i \leq l\%$, and n is the number of all predictions. Unlike both *MMRE* and *MdMRE*, for *PRED(l)*, a higher score implies better prediction accuracy.

5.3 Experiments and Results

In the experiments, we have made the following three investigations:

1. Investigation 1 aims to find out outliers and to choose optimal feature subset from the given data sets for the prediction, both outlier detection and feature subset selection are wrapper methods, and the wrapper method itself measures the MRE and MMRE implicitly, so only the number of reduced features and the number of outliers are presented.

2. Investigation 2 further explores the effectiveness of the outlier detection and feature subset selection methods. In order to provide readers a concise result, and the MMRE is the most widely used evaluation criterion, so only the values of MMRE are presented.

3. In Investigation 3, we intended to thoroughly evaluate the methods. So every possible combinations of with/without feature subset selection, with/without outlier detection for all the three methods with the five data sets are considered, and all the results including predicted efforts, the values of MRE, MMRE and PRED are presented.

5.3.1 Investigation 1

In this experiment, we intended to provide quality data for software effort prediction by means of removing outliers and choosing predictive feature subset.

For this purpose, the proposed outlier detection method OD@GRACE$^+$ and the proposed feature subset selection method FSS@GRACE$^+$ have been applied to each of the five data sets respectively. Table 7 contains the results. Table 7 shows that both features and projects have been reduced, and the extent of the reduction of features are much greater than those of projects. This is what we desired, as outliers always are extremely smaller than normal data. This means OD@GRACE$^+$ and FSS@GRACE$^+$ can efficiently detect outliers and reduce the number of features. The effectivenesses of these two methods will be justified in Investigation 2 – the next subsection.

	without FSS	with FSS	Mean
without OD	77.7	54.7	66.2
with OD	31.9	26.1	29.0
Mean	54.8	40.4	

Table 8: *MMRE* (%) for GRACE$^+$ with the Albrecht data set

5.3.2 Investigation 2

In this experiment, we intended to answer the following questions:

1. Does the proposed outlier detection method OD@GRACE$^+$ improve the performance of the proposed effort prediction method EP@GRACE$^+$?

2. Does the proposed feature subset selection method FSS@GRACE$^+$ improve the performance of the proposed effort prediction method EP@GRACE$^+$?

3. Does the combining use of the outlier detection method OD@GRACE$^+$ and the feature subset selection method FSS@GRACE$^+$ more greatly improve the performance of the proposed effort prediction method EP@GRACE$^+$?

For these purposes, for each of the five data sets, EP@GRACE$^+$ has been applied to the four types of the induced data sets: an original data set, a data set only with feature subset selection, a data set only with outlier detection, and a data set with both outlier detection and feature subset selection. Tables 8, 9, 10, 11, and 12 contain the results in terms of *MMRE* for the five data sets respectively. For each table, the last row contains the mean of *MMRE*s without/with feature subset selection regardless of the adoption of outlier detection respectively, while the last column holds the mean of *MMRE*s without/with outlier detection whether the feature subset selection was performed or not individually. These tables provide us a whole picture of whether the feature subset selection method or the outlier detection method is able to improve the software effort prediction accuracy. From them we observe that the consistent results are obtained. That is for each data set, *MMRE*s of the induced data sets with outlier detection are smaller than those of without for both with or without feature subset selection, which reveals that the proposed outlier detection method OD@GRACE$^+$ improved the performance of the proposed effort prediction method EP@GRACE$^+$ at least for the five data sets; *MMRE*s of the induced data sets with feature subset selection are smaller than those without for both with or without outlier detection, which reveals that the feature subset selection method FSS@GRACE$^+$ improved the performance of the proposed effort prediction method EP@GRACE$^+$ at least for the five data sets; *MMRE* of the induced data set with both outlier detection and feature subset selection is much smaller than those with only outlier detection or feature subset selection and without both of them, which reveals that combining use of the outlier detection method OD@GRACE$^+$ and the feature subset selection method greatly improved the prediction accuracy at least for the five data sets.

	without FSS	with FSS	Mean
without OD	33.7	25.3	29.5
with OD	31.7	24.2	27.95
Mean	32.7	24.75	

Table 9: *MMRE* (%) for GRACE$^+$ with the COCOMONASA data set

	without FSS	with FSS	Mean
without OD	63.8	55.2	59.5
with OD	60.0	49.8	54.9
Mean	61.9	52.5	

Table 10: *MMRE* (%) for GRACE$^+$ with the COCOMO81 data set

	without FSS	with FSS	Mean
without OD	55.1	44.4	49.75
with OD	44.7	41.4	43.05
Mean	49.9	42.9	

Table 11: *MMRE* (%) for GRACE$^+$ with the Desharnais data set

	without FSS	with FSS	Mean
without OD	51.6	44.0	47.8
with OD	46.1	19.6	32.85
Mean	48.85	31.8	

Table 12: *MMRE* (%) for GRACE$^+$ with the Kemerer data set

5.3.3 Investigation 3

In this experiment, we intended to answer the following questions:

1. Does the proposed effort prediction method GRACE$^+$ outperform SWR?
2. Does the proposed effort prediction method GRACE$^+$ outperform Analogy?
3. Does the proposed outlier detection method OD@GRACE$^+$ improve the performance of both SWR and Analogy?

For these purposes, for each of the five data sets, GRACE$^+$, SWR, and Analogy respectively have been applied to the two types of the induced data sets: a data set with outlier detection and another without. We present both the detailed and summarized results.

Figs. [7] 2, 3, 4, 5, and 6 are the scatter plots, which provide us a whole picture of the real software project efforts, the predicted efforts, and the relationships between them, of the respective methods with the five data sets. From these figures we observe a common pattern that is for all the three methods, there is a positive correlation between the real software efforts and the predicted results. This reveals that small values of the predicted effort correspond to small values of the real software effort, large values of the predicted effort correspond to large values of the real software effort, and this is what we desired. At the same time, we also observe that for each of the three methods, the scatters are different; specifically, in 8 out of 10 cases, the scatters of GRACE$^+$ are smaller than both of the other two methods. This reveals that the prediction accuracies of the different methods are different, and GRACE$^+$ is more accurate than the other two methods in 8 out of 10 cases.

More importantly, Figs. 2, 3, 4, 5, and 6 also reveal a statistical condition referred to as heteroscedasticity, which is non-constant variation in the predicted results over the values of the real software effort. We observe that small values of the real software effort yield small scatter in the predictions while large values of the real software effort result in large scatter in predictions. It seems that it is harder to precisely predict the effort of a large software project than that of a small one. Fortunately, we also observe that the heteroscedasticity of GRACE$^+$ is weaker than or equal to those of the other two methods. This means generally GRACE$^+$ gives better predictions regardless of the size of a software project.

Figs. 7, 8, 9, 10, and 11 contain the detailed prediction results of the respective methods with the five data sets in terms of boxplots of *MRE*.

The boxplot is a type of graph that is used to show the shape of a distribution, its central value, and variability. We used boxplots to explore the spreads of the magnitude of relative error (*MRE*) of prediction accuracy for GRACE$^+$, SWR, and Analogy with the Albrecht, COCOMONASA, COCOMO81, Kemerer, and Desharnais data sets respectively.

Fig. 7 contains the results for the three methods with the Albrecht data set. From it we observe that before removing outliers: 1) GRACE$^+$ has the smallest lower and upper whiskers, which means both the best and the worst prediction results of GRACE$^+$ are more accurate than those of Analogy and SWR, also

[7] Where *MethodName*+ means using the *MethodName* method with outlier detection otherwise without.

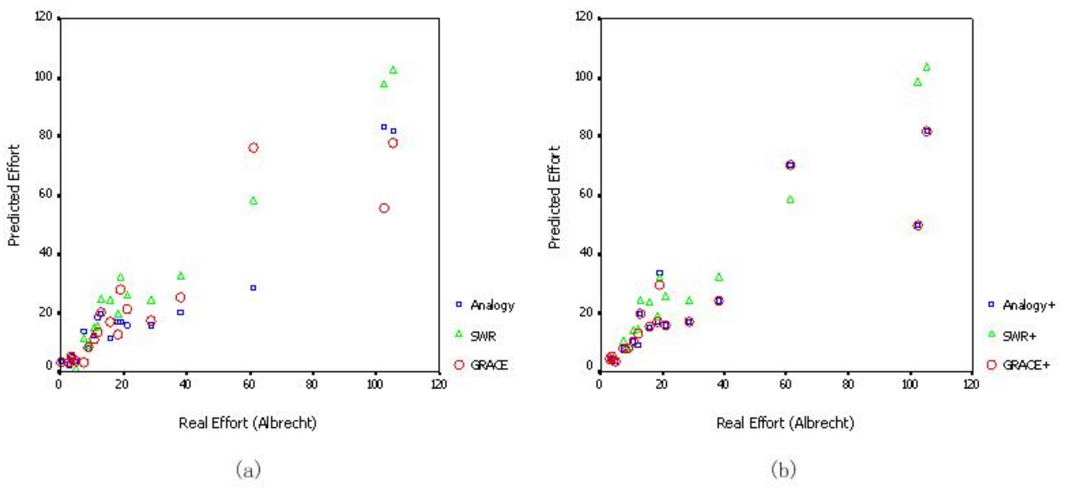

Figure 2: The predicted vs. real software effort for GRACE$^+$, Analogy, and SWR with the Albrecht data set

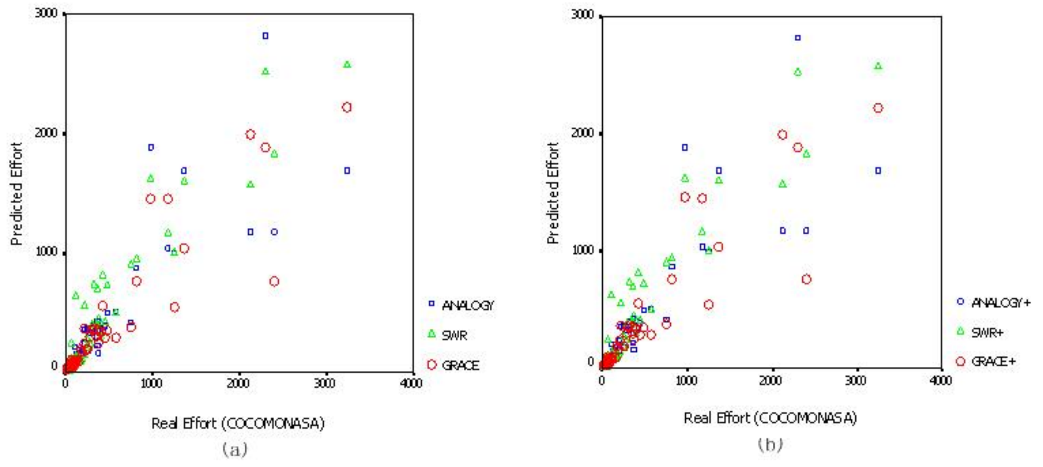

Figure 3: The predicted vs. real software effort for GRACE$^+$, Analogy, and SWR with the COCOMONASA data set

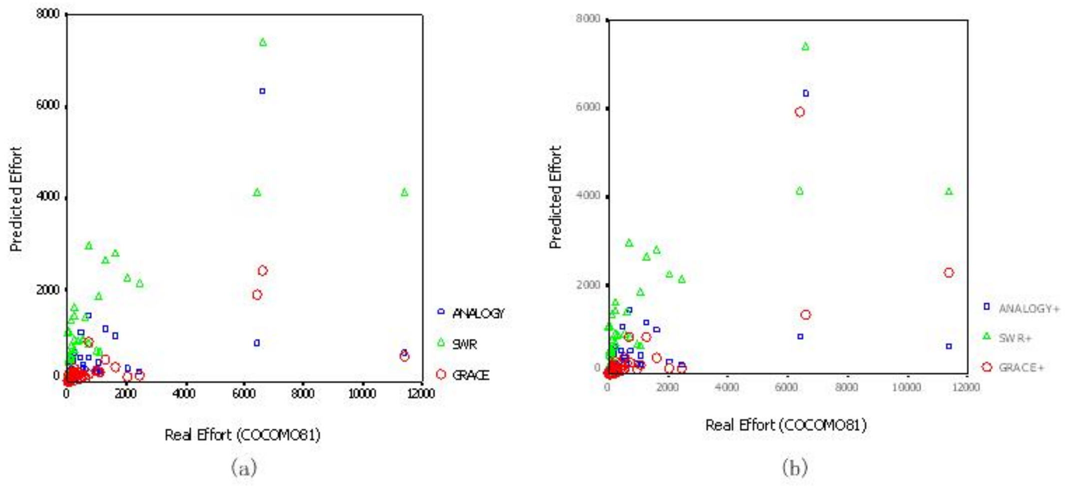

Figure 4: The predicted vs. real software effort for GRACE$^+$, Analogy, and SWR with the COCOMO81 data set

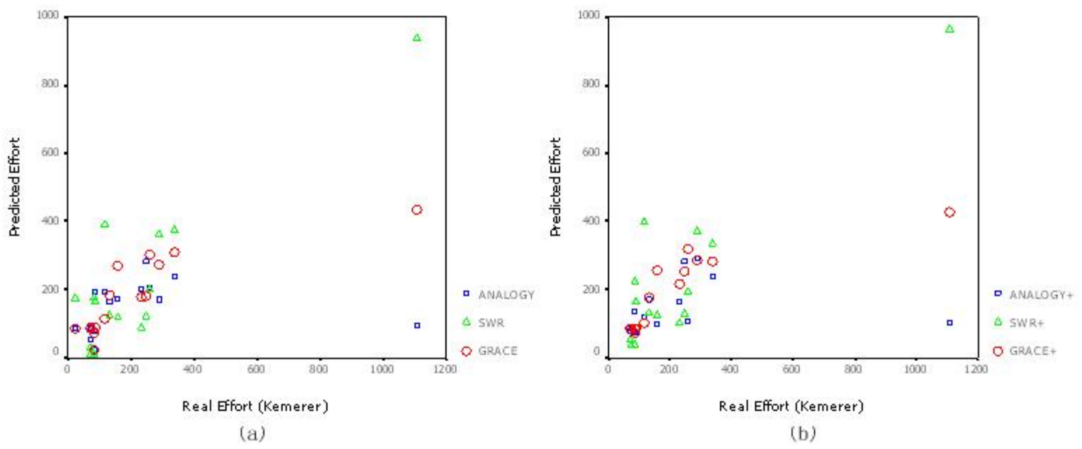

Figure 5: The predicted vs. real software effort for GRACE$^+$, Analogy, and SWR with the Kemerer data set

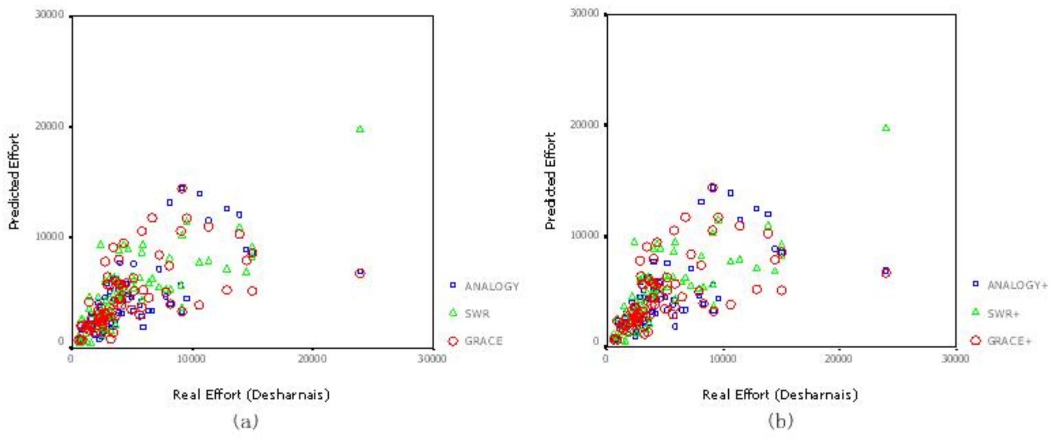

Figure 6: The predicted vs. real software effort for GRACE$^+$, Analogy, and SWR with the Desharnais data set

indicates reduced variability of *MRE*s; 2) GRACE$^+$ has the smallest upper quartile, which reveals that at least 75% the predictions of GRACE$^+$ are more accurate than those of Analogy and SWR. We also observe that after removing outliers: 1) GRACE$^+$ has the smallest lower and upper whiskers, which means both the best and the worst prediction results of GRACE$^+$ are more accurate than those of Analogy and SWR, also indicates reduced variability of *MRE*s; 2) The upper quartile of GRACE$^+$ is much smaller than that of SWR but greater than that of Analogy, which reveals that at least 75% the predictions of GRACE$^+$ are much more accurate than that of SWR but less accruate than that of Analogy. Comparing the results with and without outlier detection, we found that the proposed outlier detection method has improved the results of both GARCE and Analogy, but just reduced the median of SWR.

Fig. 8 contains the results for the three methods with the COCOMONASA data set. From it we observe that before removing outliers: 1) The upper whiskers of GRACE$^+$ and Analogy are much smaller than that of SWR, which means the worst prediction results of GRACE$^+$ and Analogy are more accurate than that of SWR, also indicates reduced variability of *MRE*s for both GRACE$^+$ and Analogy; 2) GRACE$^+$ has the smallest upper quartile, which reveals that at least 75% the predictions of GRACE$^+$ are more accurate than those of Analogy and SWR; 3) GRACE$^+$ has only one outlier and has no extreme. We also observe that after removing outliers: 1) GRACE$^+$ has the smallest upper whisker, which means the worst prediction result of GRACE$^+$ is more accurate than those of Analogy and SWR, also indicates reduced variability of *MRE*s; 2) GRACE$^+$ has the smallest upper quartile, which reveals that at least 75% the predictions of GRACE$^+$ are more accurate than those of Analogy and SWR; 3) GRACE$^+$ has only one outlier and others either have outliers or outliers and extremes. Comparing the results with and without outlier detection, we found that for GRACE$^+$, the only outlier's value has been greatly reduced; for Analogy, the extreme has been removed; for SWR, one extreme and severial outliers have been removed. This indicates that the proposed outlier detection method has improved the results of all the three methods.

Fig. 9[8] contains the results for the three methods with the COCOMO81 data set. From it we observe that both before and after removing outliers: 1) GRACE$^+$ has the smallest upper whisker, which means the worst prediction result of GRACE$^+$ is much more accurate than those of SWR and Analogy, also means reduced variability of *MRE*s; 2) GRACE$^+$ has the smallest upper quartile, which reveals that at least 75% the predictions of GRACE$^+$ are more accurate than those of Analogy and SWR; 3) before removing outliers, GRACE$^+$ has only one extreme value and this is much smaller than those of Analogy and SWR; after removing outliers, GRACE$^+$ has no extreme value unlike Analogy or SWR. Comparing the results with and without outlier detection, we find that for GRACE$^+$, the range of *MRE*s has beed greatly reduced and the only extreme has been removed; for Analogy, there is no obvious changes to the results; for SWR, the range of *MRE*s has been extended. This indicates that the proposed outlier detection method works well for GRACE$^+$.

Fig. 10 contains the results for the three methods with the Kemerer data set. From it we observe that both before and after removing outliers: 1) GRACE$^+$ has the smallest upper whisker, which means the worst prediction result of GRACE$^+$ is much more accurate than those of SWR and Analogy, also means reduced variability of *MRE*s; 2) GRACE$^+$ has the smallest upper quartile, which reveals that at least 75% the predictions of GRACE$^+$ are more accurate than those of Analogy and SWR; 3) after removing outliers, the box of GRACE$^+$ overlays the lower whisker. This reveals that the data are probably skewed towards the lower end of the scale, and also means more accurate results. Comparing the results with and without outlier detection, we find that for all three methods, the ranges of *MRE*s and the upper quartiles have been greatly reduced, there are still outlier(s) or extreme(s) but the corresponding values have been greatly reduced. This indicates that the proposed outlier detection method works well for all three methods.

Fig. 11 contains the results for the three methods with the Desharnais data set. From it we observe that both before and after removing outliers: 1) Analogy has the smallest upper whisker, which means the worst prediction result of Analogy is more accurate than those of SWR and GRACE$^+$, also means reduced variability of *MRE*s; 2) Analogy has the smallest upper quartile, which reveals that at least 75% the predictions of Analogy are more accurate than those of GRACE$^+$ and SWR; 3) Grace's range of non-outlier *MRE*s is between those of Analogy and SWR but it has the smallest median which means that at least half of the predictions of GRACE$^+$ are more accurate than other methods; 4) similar sized boxes mean similar variability of the non-outlier *MRE*s; 5) Analogy has the biggest median, which means that at least half of its predictions are less accurate than other methods. Comparing the results with and without outlier detection, we find that for both GRACE$^+$ and SWR, the number of outliers has been reduced; for SWR, the range of the non-outlier *MRE*s has been reduced. This indicates that the proposed outlier detection method works well for all three methods.

To summarize, the boxplots of *MRE*s of prediction accuracy for GRACE$^+$, Analogy and SWR with five data sets show GRACE$^+$ outperformed both Analogy and SWR for 4 out of 5 data sets.

[8]The scale has been truncated so the even worse outliers for SWR are not visible.

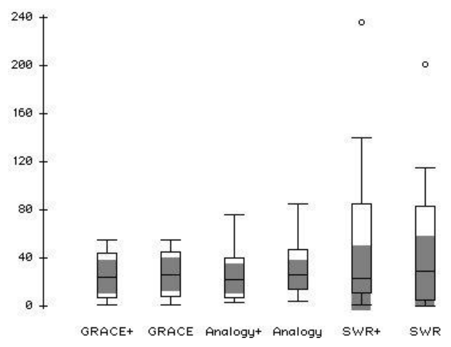

Figure 7: $MRE(\%)$ for GRACE$^+$, Analogy, and SWR with the Albrecht data set

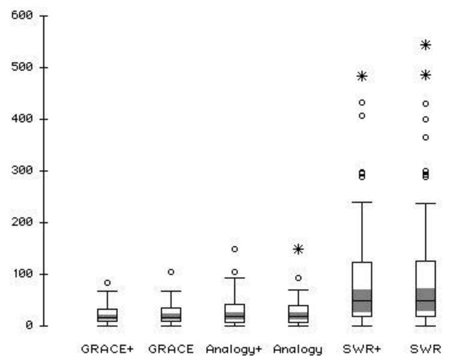

Figure 8: $MRE(\%)$ for GRACE$^+$, Analogy, and SWR with the COCOMONASA data set

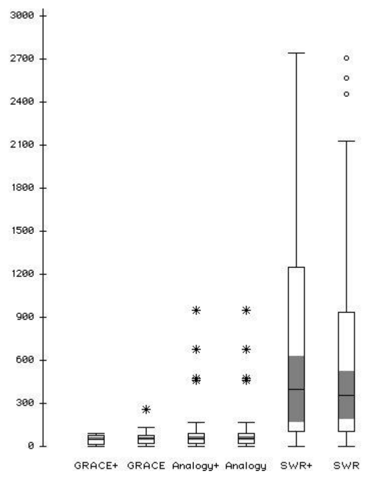

Figure 9: $MRE(\%)$ for GRACE$^+$, Analogy, and SWR with the COCOMO81 data set

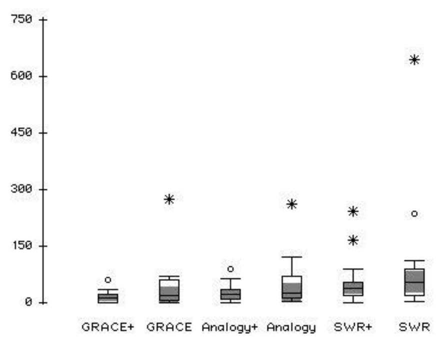

Figure 10: MRE(%) for GRACE$^+$, Analogy, and SWR with the Kemerer data set

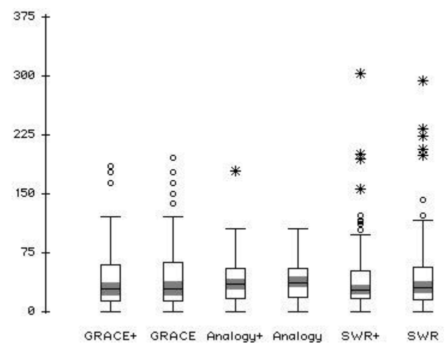

Figure 11: MRE(%) for GRACE$^+$, Analogy, and SWR with the Desharnais data set

Tables 13, 14, and 15 contain the summarized prediction results of respective methods with the five data sets in terms of *MMRE*, *MdMRE*, and *PRED(25)* respectively.

Table 13 contains the *MMRE*s for the three methods with the five data sets. From it we observe that for the three methods with and without outlier detection, the *MMRE*s of GRACE$^+$ are much smaller than those of SWR for all five data sets, and also smaller than those of Analogy except for the Desharnais data set. We also observe that for all three methods, the *MMRE*s with the proposed outlier detection method are smaller than those without except for Analogy in which it is 0.6% greater with the COCOMONASA data set. This reveals that in terms of *MMRE*, the answers to the questions are positive. That is, GRACE$^+$ tends to allow more accurate predictions – in terms of *MMRE* – than both SWR and Analogy; at the same time, OD@GRACE$^+$ improved prediction accuracy for all three methods at least with the five data sets.

Table 14 contains the *MdMRE*s for the three methods with the five data sets. From it we observe that: 1) for the three methods without outlier detection, the *MdMRE*s of GRACE$^+$ are smaller than those of SWR and Analogy for all five data sets; 2) for the three methods with outlier detection, the *MdMRE*s of GRACE$^+$ are smaller than those of Analogy except for 1.2% greater with the Albrecht data set and those of SWR except 1.1% and 0.4% greater with the Albrecht data set and the Desharnais data set respectively; 3) for all three methods, the *MdMRE*s with the proposed outlier detection method are smaller than those without except for Analogy with the Kemerer data set which is 2.5% greater. This reveals that in terms of *MdMRE*, the answers to the questions are positive. That is, GRACE$^+$ tends to allow more accurate predicions – in

Data Set		*MMRE*(%) of		
		GRACE$^+$	SWR	Analogy
Albrecht	without OD	54.7	132.2	61.8
	with OD	26.1	54.1	26.6
COCOMONASA	without OD	25.3	102.6	28.3
	with OD	24.2	91.4	28.9
COCOMO81	without OD	55.2	1540.8	97.4
	with OD	49.8	1503.2	96.6
Desharnais	without OD	44.4	51.3	39.4
	with OD	41.4	46.5	39.0
Kemerer	without OD	44.0	102.6	54.2
	with OD	19.6	58.5	29.2

Table 13: *MMRE*(%) for GRACE$^+$, SWR, and Analogy with different data sets

terms of *MdMRE* – than both SWR and Analogy and OD@GRACE$^+$ tends to improve predcition accuracy for all three methods.

Table 15 contains the *PRED(25)*s for the three methods with the five data sets. From it we observe that *PRED(25)* values are more mixed. This may in part be due to the arbitrary effect of setting the threshold to 25%[9]. Specifically, 1) the GRACE$^+$'s *PRED(25)*s are greater than those of SWR except for the Albrecht data set which was equal to the latter with outliter detection and 5.3% smaller without outlier detection; 2) the GRACE$^+$'s *PRED(25)*s are greater than those of Analogy for 3 out of 5 data sets for both with and without outlier detection; 3) for GRACE$^+$, the *PRED(25)*s with outlier detection are greater than those of without for 3 out of 5 data sets and 0.8% and 0.2% smaller for other two data sets; for both SWR and Analogy, the *PRED(25)*s with outlier detection are greater than those of without for three out of five data sets. This reveals that GRACE$^+$ generally outperformed Analogy and SWR and the proposed outlier detection method improved *PRED(25)* for the three methods with the majority of data sets we used.

For the purpose of more formally determining whether the accuracy improvements of GRACE$^+$ are statistical significant compared with SWR and Analogy, we used the Wilcoxon test to compare sample medians of the corresponding accuracies. For all the tests, the null hypotheses are that there is no difference with ($\alpha = 0.05$). We have alternate hypotheses as follows:

- GRACE$^+$ is more accurate than both SWR and Analogy with the Albrecht data set with/without outlier detection;
- GRACE$^+$ is more accurate than both SWR and Analogy with the COCOMONASA data set with/without outlier detection;

[9]The reason *PRED(25)* is used is because it is widely used by many researchers (e.g., Shepperd and Schofield [42], Jeffery et al. [20], Boetticher [4], Wittig and Finnie [53], Saliu et al. [40], Walkerden and Jeffery [48], etc.), so using this value can make us directly compare our results with the published results.

Data Set		*MdMRE*(%) of		
		GRACE$^+$	SWR	Analogy
Albrecht	without OD	26.7	34.8	27.9
	with OD	24.2	23.1	23.0
COCOMONASA	without OD	18.1	50.0	20.2
	with OD	17.0	49.8	20.2
COCOMO81	without OD	55.6	445.1	59.8
	with OD	55.2	402.5	58.6
Desharnais	without OD	29.8	31.2	38.0
	with OD	29.2	28.8	36.8
Kemerer	without OD	23.2	59.7	26.1
	with OD	13.8	47.9	28.6

Table 14: *MdMRE*(%) for GRACE$^+$, SWR, and Analogy with different data sets

Data Set		*PRED(25)*(%) of		
		GRACE$^+$	SWR	Analogy
Albrecht	without OD	42.1	47.4	47.4
	with OD	50.0	50.0	55.6
COCOMONASA	without OD	58.3	35.0	65.0
	with OD	60.3	34.5	63.8
COCOMO81	without OD	30.2	7.9	22.6
	with OD	29.0	8.1	22.2
Desharnais	without OD	45.5	39.0	31.2
	with OD	45.3	41.9	35.1
Kemerer	without OD	53.3	33.3	46.7
	with OD	78.6	28.6	50.0

Table 15: *PRED(25)*(%) for GRACE$^+$, SWR, and Analogy with different data sets

Data Set		p Value of $MREs$	
		Ha: GRACE$^+$ < SWR	Ha: GRACE$^+$ < Analogy
Albrecht	without OD	0.0465	0.0474
	with OD	0.0311	0.4262
COCOMONASA	without OD	0.0001	0.0486
	with OD	0.0001	0.0467
COCOMO81	without OD	0.0001	0.0264
	with OD	0.0001	0.0183
Kemerer	without OD	0.0289	0.0235
	with OD	0.0466	0.0460
Data Set		Ha: GRACE$^+$ < SWR	Ha: GRACE$^+$ > Analogy
Desharnais	without OD	0.0389	0.1724
	with OD	0.0235	0.2882

Table 16: p value for GRACE$^+$, SWR, and Analogy with different data sets

- GRACE$^+$ is more accurate than both SWR and Analogy with the COCOMO81 data set with/without outlier detection;
- GRACE$^+$ is more accurate than both SWR and Analogy with the Kemerer data set with/without outlier detection;
- GRACE$^+$ is more accurate than SWR with the Desharnais data set with/without outlier detection;
- Analogy is more accurate than GRACE$^+$ with the Desharnais data set with/without outlier detection.

Table 16 contains the results of the Wilcoxon test. From the table we observe that all the hypotheses are accepted except for "GRACE$^+$ is more accurate than Analogy with the Albrecht data set with outlier detection" and "Analogy is more accurate than GRACE$^+$ with the Desharnais data set with/without outlier detection" are rejected. This reveals: 1) in 4 out of 5 data sets, the accuracy improvements of GRACE$^+$ with/without outlier detection are statistically significant except for the Albrecht data set with outlier detection; 2) the accuracy improvements of Analogy with the Desharnais data set with/without outlier detection are not statistically significant.

On the other hand, the use of the same data set and the same experimental method allow us to compare GRACE$^+$ with NW GRA [16], LW GRA [16], ANN [41, 18], CART [46, 18], GRA [18], and GP [10] methods.

Table 17 contains the published results of *MMRE*. It shows that GRA based methods, GRACE$^+$, NW GRA, LW GRA, and GRA, all outperform others with all the four data sets. Of the GRA based methods, compared with NW GRA, the *MMRE* has been improved by GRACE$^+$ by 37% and 69.8% respectively with the COCOMO81 data set and the Kemerer data set; Compared with LW GRA, the *MMRE* has been improved by GRACE$^+$ by 18.4% and 66.4% respectively with the COCOMO81 data set and the Kemerer

Data Set	*MMRE*(%) of						
	GRACE$^+$	NW GRA [16]	LW GRA [16]	ANN [41, 18]	CART [46, 18, 6]	GRA [18]	GP [10]
Albrecht	26.1	N.A.	N.A.	86	76	31	N.A.
COCOMO81	49.8	79	61	143	125	69	N.A.
Desharnais	41.4	N.A.	N.A.	N.A.	N.A.	N.A.	44.55
Kemerer	19.6	65	58.3	70	364,94	N.A.	N.A.

Table 17: Comparison of *MMRE*s with published results

Data Set	*PRED(25)*(%) of						
	GRACE$^+$	NW GRA [16]	LW GRA [16]	ANN [41, 18]	CART [46, 18]	GRA [18]	GP [10]
Albrecht	50	N.A.	N.A.	21	26	48	N.A.
COCOMO81	29	14.2	25.3	11	25	38	N.A.
Desharnais	45.3	N.A.	N.A.	N.A.	N.A.	N.A.	23.3
Kemerer	78.6	20	33.3	20	20	N.A.	N.A.

Table 18: Comparison of *PRED(25)*s with published results

data set; Compared with GRA, the *MMRE* has been improved by GRACE$^+$ by 15.8% and 27.8% respectively with the Albreht data set and the COCOMO81 data set.

Table 18 contains the published results of *PRED(25)*. It shows that GRA based methods, GRACE$^+$, NW GRA, LW GRA, and GRA, all outperform others with all the four data sets. Of the GRA based methods, compared with NW GRA, the *PRED(25)* has been improved by GRACE$^+$ by 104.2% and 293% respectively with the COCOMO81 data set and the Kemerer data set; Compared with LW GRA, the *PRED(25)* has been improved by GRACE$^+$ by 14.6% and 136% respectively with the COCOMO81 data set and the Kemerer data set; Compared with GRA, the *MMRE* has been improved by GRACE$^+$ by -4.2% and 23.7% respectively with the Albreht data set and the COCOMO81 data set.

To summarize, generally, GRACE$^+$ is more accurate than the other compared methods at least for the given data sets excepting a 4.2% *PRED(25)* smaller than the GRA method with the Albreht data set. On other hand, the proposed outlier detection method improved the performance of GRACE$^+$, SWR and Analogy.

5.3.4 Discussion

The empirical results show that the proposed GRACE$^+$ method leads to more accurate effort estimates than the competing approaches, such as Analogy and SWR. Actually, the same conclusion can be obtained from the following analyses: 1) GRACE$^+$ uses L1 norm metric to compute the distances among different projects while other methods, such as Analogy and SWR, employ L2 norm metric that is sensitive to outliers [25].

Thus, GRACE$^+$ is more suitable for the data sets with outliers; 2) Further, GRACE$^+$ uses a unified L1 norm to compute the distances among any types of features, while Analogy employs L2 norm when computing distances among continuous features and uses Hamming distance (one kind of L1 norm) for nominal features. Compared to an inconsistent way to compute distance, a unified distance metrics is more reasonable and favorable. Moreover, software project data sets always contain both continuous and nominal features. Thus, GRACE$^+$ is more appropriate to deal with software project data; 3) GRACE$^+$ is based on GRA that was designed to utilize only a few known data to establish an analysis model [12], while Analogy is a kind of k-NN method that requires large data sets [35]. Meanwhile, smallness is one of the major characteristics of the software project data sets [38]. This makes k-NN is not best for software project data; 4) GRACE$^+$ is based on GRA that has no any data distribution requirement, while machine learning methods assume that data come from a multivariate normal distribution, which may not be the case for software project data.

6 Conclusions

In this paper, we have proposed a novel approach of using Grey Relational Analysis of Grey System Theory to address outlier detection, feature subset selection and software effort prediction at an early stage of a software development process.

Using five publicly available data sets, we have compared the proposed method with prediction models based on stepwise regression and Analogy, and when available, also with NW GRA [16], LW GRA [16], ANN [41, 18], CART [46, 18], GRA [18], and GP [10] methods. The results show that the proposed method outperformed 1) stepwise regression and Analogy in terms of *MMRE*, *MdMRE*, and *PRED(25)* for the majority (4 out of 5) of data sets we used; 2) other methods in terms of *MMRE* and *PRED(25)* excepting for a 4.2% *PRED(25)* smaller than GRA with one data set. The results also show that the proposed outlier detection method can not only improve the performance of the proposed effort prediction method but also the performance of SWR and Analogy. This is very encouraging and indicates that the method has considerable potential.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under grant 61070006.

References

[1] Albrecht, A. and Gaffney, J. “Software Function, Source Lines of Code, and Development Effort Prediction”. *IEEE Transactions on Software Engineering*, Vol.9, No.6, pp.639–648, 1983.

[2] Boehm, B.W., “Software Engineering Economics”. *IEEE Transactions on Software Engineering*, Vol.10, No.1, pp.4–21, 1984.

[3] Boehm, B.W., *Software Engineering Economics*, Prentice-Hall, 1981.

[4] Boetticher, G. When will it be done? the 300 billion dollar question, machine learner answers. IEEE Intelligent Systems, Vol. 18, No. 3, pp. 48-50, 2003.

[5] Breunig, M. M., Kriegel, H. and Raymond T. Ng and Sander, J., "LOF: Identifying Density-Based Local Outliers". *ACM SIGMOD Record*, Vol.29, No.2, pp.93–104, 2000.

[6] Briand, L., Basili, V., and Thomas, W., "A Pattern Recognition Approach for Software Engineering Data Analysis". *IEEE Transactions on Software Engineering*, Vol.18, No.11, pp.931–942, 1992.

[7] Briand, L., T. Langley and I. Wieczorek. "Using the European Space Agency data set: a replicated assessment and comparison of common software cost modeling techniques". *22ndInternational Conference on Software Engineering*, pp.377–386, Limerick, Ireland: Computer Society Press, 2000.

[8] Briand, L.C. and Wieczorek, I.,"Resource Modeling in Software Engineering," in *Encyclopaedia of Software Engineering*, J. J. Marciniak, Ed., 2nd ed. New York: John Wiley, 2002.

[9] Cheung, R. and Eisenstein, B., "Feature selection via dynamic programming for text-independent speaker identification". *IEEE Transactions on Acoust.,Speech and Signal Processing*, Vol. ASSP-26, No.5, pp.397-403, 1978.

[10] Burgess, C.J. and Lefley, M., "Can genetic programming improve software effort estimation? A comparative evaluation". *Information and Software Technology*, Vol.43, No.14, pp.863–873, 2001.

[11] Deng, J. L., "Control problems of grey system". *System and Control Letters*, Vol. 1, pp.288–294, 1982.

[12] Deng, J. L., "Properties of relational space for grey system". In *Essential Topics on Grey System Theory and Applications* (J. L. Deng, ed.), pp.1-13. Beijing: China Ocean, 1988.

[13] Desharnais, J. M., "Analyse statistique de la productivitie des projets informatique a partie de la technique des point des fonction". *Masters Thesis*, University of Montreal, 1989.

[14] Finnie, G.R and Wittig, G.E., "A comparison of software effort technique: using function points with neural networks, case based reasoning and regression models". *Journal Systems and Software*, Vol.39, pp.281–289, 1997.

[15] Hsu, C. J., Huang, C. Y. and Hung, T. Y., *A Study of Applying Grey Relational Analysis to Software Effort Estimation*, CD-ROM Proceedings of the 36th International Conference on Computers and Industrial Engineering (ICCIE 2006), June 2006, Taipei, Taiwan.

[16] Hsu, C. J. and Huang, C. Y., *Improving Effort Estimation Accuracy by Weighted Grey Relational Analysis During Software Development Process*, Proceedings of the 14th Asia-Pacific Software Engineering Conference (APSEC 2007), pp. 534-541, Nagoya, Japan, Dec. 2007.

[17] Huang, S. J., Huang, C. L., "Control of an inverted pendulum using grey prediction model". *IEEE Transactions on Industry Applications*, Vol.36, No.2, pp.452–458, 2000.

[18] Huang, S. J., Chiu, N. H., Chen, L. W, *Integration of the grey relational analysis with genetic algorithm for software effort estimation*, European Journal of Operational Research, Volume 188, Issue 3, 1 August 2008, Pages 898-909

[19] Jain, A. and Zongker, D., "Feature Selection: Evaluation, application and small sample performance". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.19, No.2, pp.153–158, 1997.

[20] Jeffery, R. , Ruhe, M., and Wieczorek, I. A Comparative study of two software development cost modeling techniques using multi-organizational and companyspecific data. Information and Software Technology, Vol.42, pp.1009-1016, 2000.

[21] Jiang, B.C., Tasi, S. L., and Wang, C. C., "Machine vision-based gray relational theory applied to IC marking inspection". *IEEE Transactions on Semiconductor Manufacturing*, Vol.15, No.4, pp.531–539, 2002.

[22] Jørgensen, M., Indahl, U., and Sjøberg, D.,"Software effort estimation by analogy and 'regression toward the mean' ". *Journal of Systems and Software*, Vol.68, No.3, pp.253–262, 2003.

[23] Jou, J. M., Chen, P. Y., and Sun, J. M., "The gray prediction search algorithm for block motion estimation". *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.9, No.6, pp.843–848, 1999.

[24] Kadoda, G., Cartwright, M. and Shepperd, M.J., "Issues on the effective use of CBR technology for software project prediction", *4th Intl. Conf. on Case Based Reasoning*, Vancouver, 2001.

[25] Ke, Q. and Kanade, T. "Robust Subspace Computation Using L1 Norm,*Tech. Report CMU-CS-03-172*, Computer Science Department, Carnegie Mellon University, 2003.

[26] Kemerer, C.F., "An empirical validation of software cost estimation models". *Comm. ACM*, Vol.30, No.5, pp.416–429,1987.

[27] Khotanzad, A. and Kashyap, R., "Feature Selection for Texture Recognition Based on Image Synthesis". *IEEE Transactions on Systems, Man, and Cybernetics*, Vol.17, pp.1087-1095, 1987.

[28] Knorr, E. M. and Ng, R. "Algorithms for Mining Distance-Based Outliers in Large Datasets". *Proc. 24th Int. Conf. Very Large Data Bases*, pp.392–403, 1998.

[29] Kohavi, R. and G. H. John, "Wrappers for feature selection for machine learning." *Artificial Intelligence* Vol. 97, Nos. 1-2, pp. 273-324.

[30] Li, P. , Tan, T. C. and Lee, J. Y., "Grey relational analysis of amine Statistics of mild steel corrosion in acids". *Corrosion* Vol.53, pp.186-194, 1997.

[31] Liu, S., Lin, y., *Grey Information: Theory and Practical Applications* , 1st edition, Springer. 2006.

[32] Luo, R. C., Chen, T. M., and Su, K. L., “Target tracking using a hierarchical grey-fuzzy motion decision-making method”. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, Vol.31, No.3, pp.179–186, 2001.

[33] Mair, C., Kadoda, G., Lefley, M., Phalp, K., Schofield, C., Shepperd, M. and Webster, S., “An investigation of machine learning based prediction systems”. *Journal of Systems and Software.* Vol.53, No.1, pp.23–29, 2000.

[34] Menzies, T., Port, D., Chen, Z., Hihn, J. and Stukes, S., ”Validation Methods for Calibrating Software Effort Models”. *The 27th International Conference on Software Engineering.* St Louis Missouri, USA, 15–21 May, 2005.

[35] Mitchell, T. Machine Learning, McGraw Hill, 1997.

[36] Moløkken, K. and Jørgensen, M., “A review of surveys on software effort estimation”, *2nd Intl. Symp. on Empirical Software Engineering*, pp.223-230: IEEE Computer Society, 2003.

[37] Mukhopadhyay, T., Vicinanza, S.S., and Prietula, M.j. “Examining the feasibility of a case-based reasoning model for software effort estimation.” *MIS Quarterly*, Vol.16, pp.155–171, june, 1992.

[38] Myrtveit, I., Stensrud E. and Olsson, U.H., “Analyzing data sets with missing data: An empirical evaluation of imputation methods and likelihood-based methods”, *IEEE Transactions on Software Engineering*, Vol.27, No.11, pp999-1013, 2001.

[39] Putnam, L. H., “A general empirical solution to the macro software sizing and estimation problem”. *IEEE Transactions on Software Engineering*, Vol.4, No.4, pp.345–381,1978.

[40] Saliu, M.O., Ahmed, M., and AlGhamdi, J. Towards adaptive soft computing based software effort prediction. IEEE Annual Meeting of the Fuzzy Information, 2004. Processing NAFIPS ’04. Vol. 1, pp.16 - 21, 27-30 June 2004.

[41] Samson, B., Ellison, D., and Dugard, P., “Software coast estimation using an Albus Perceptron (CMAC)”. *Information and Software Technology*, Vol.39, Nos.1/2, 1997.

[42] Shepperd, M. and Schofield, C., “Estimating Software Project Effort Using Analogies”. *IEEE Transactions on Software Engineering*, Vol.23, No.12, pp.736–743, 1997.

[43] Siedelecki, W. and Skalansky, J., “On automatic feature selection”. *International Journal of Pattern Recognition and Artificial Intelligence*, Vol.2, No.2, pp.197-220, 1988.

[44] Song, Q. and Shen, J., “A Web Document Clustering Algorithm Based on Mutual Nearest Neighborhood”. *Journal of Computer Research and Development*, Vol.38, No.7, pp.30-34, 2001.

[45] Song, Q., Shepperd, M. and Mair, C., “ Using Grey Relational Analysis to Predict Software Effort with Small Data Sets”. *Proceedings of 11th IEEE International Software Metrics Symposium (Metrics 2005)*, 19-22 September, 2005, Como, Italy.

[46] Srinivasan, K. and Fisher, D., "Machine Learning Approaches to Estimating Software Development Effort". *IEEE Transactions on Software Engineering*, Vol.21, No.2, pp.126-137, 1995.

[47] Su, S. L., Su, Y. C., Huang, J. F., "Grey-based power control for DS-CDMA cellular mobile systems". *IEEE Transactions on Vehicular Technology*, Vol.49, No.6, pp.2081–2088, 2000.

[48] Walkerden, F. and Jeffery, V. An Empirical Study of Analogy-based Software Effort Estimation. Empirical Software Engineering, Vol. 4, No.2, pp.135-158, 1999.

[49] Wang Y, Song QB, Stephen MacDonell, Martin Shepperd, and Shen JY. Integrate the GM(1,1) and Verhulst Models to Predict Software Stage-Effort. IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews, Vol.39, No.6, 2009.

[50] Wang, Y. F., "On-demand forecasting of stock prices using a real-time predictor". *IEEE Transactions on Knowledge and Data Engineering*, Vol.15, No.4, pp.1033–1037, 2003.

[51] Wang, X., Nie, H., Lee, J. and Re, J., "Application of Grey System Models in the Agricultural Economy "(in Chinese), pp.1-21. Wuhan: Huazhong University of Science andTechnology, 1989.

[52] Wittig, G.E. and Finnie, G.R, "Using artificial neural networks and function points to estimate 4GL software development effort". *Australian Journal of Information Systems*,Vol.1, No.2, pp.87-94, 1994.

[53] Wittig, G. and Finnie, G. Estimating software development effort with connectionist models.Information and Software Technology, Vol.39, No.7, pp.469-476,1997.

[54] Wu, J.H. and Chen, C. B., "An alternative form for Grey Relational Grade". *Journal of Grey System*, Vol.11, No.1, pp.7–11,1999.

[55] Wu, H. S., Deng, J. L., and Win, K. L., *Introduction to Grey Theory* (in Chinese). Taipei, Taiwan: Gao-Li Co., 1996.