Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 188 (2008) 898–909
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Computing, Artificial Intelligence and Information Management

# Integration of the grey relational analysis with genetic algorithm for software effort estimation

Sun-Jen Huang [a,*], Nan-Hsing Chiu [b], Li-Wei Chen [a]

[a] Department of Information Management, National Taiwan University of Science and Technology, Taiwan, ROC
[b] Department of Information Management, Ching Yun University, Taiwan, ROC

Received 12 August 2005; accepted 2 July 2007
Available online 15 August 2007

**Abstract**

Accurate estimates of efforts in software development are necessary in project management practices. Project managers or domain experts usually conduct software effort estimation using their experience; hence, subjective or implicit estimates occur frequently. As most software projects have incomplete information and uncertain relations between effort drivers and the required development effort, the grey relational analysis (GRA) method has been applied in building a formal software effort estimation model for this study. The GRA in the grey system theory is a problem-solving method that is used when dealing with similarity measures of complex relations. This paper examines the potentials of the software effort estimation model by integrating 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. Experimental results show that the software effort estimation using an integration of the GRA with GA method presents more precise estimates over the results using the case-based reasoning (CBR), classification and regression trees (CART), and artificial neural networks (ANN) methods.


*Keywords:* Project management; Software effort estimation; Grey relational analysis; Genetic algorithm

## 1. Introduction

One of the most important tasks of a software project manager is to produce accurate time and effort estimates required for the completion of a software development or maintenance project. Reliable and accurate estimates of software development effort are critical to the performance of the project monitor as well as the management of the project portfolio risk [22]. Improving the accuracy of the software effort estimates in the early stages of the project life cycle facilitates a more effective plan and control of the budget and further improves the quality of the final product in the software development project.

Software effort estimation using expert opinions is one of the well-known methods [21]. Estimates using expert judgment can be produced easily and without the need of complicated tools or techniques. It is widely used in software effort estimation practices, but the process of deriving an estimate is implicit and is therefore difficult to repeat. A subjective

* Corresponding author. Tel.: +886 2 2737 6779; fax: +886 2 2737 6777.
*E-mail address:* huangsj@mail.ntust.edu.tw (S.-J. Huang).

0377-2217/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2007.07.002

estimate of software development efforts also occurs frequently. The amount of unacceptable accuracies of estimates that has been observed reveals the problems of expert judgment [29]. As a consequence, the need for a formal modeling technique has arisen.

More recently, attention has been turned to a variety of machine-learning methods such as artificial neural networks [15], analogy [28] or rule induction [30] for software effort estimation. These methods have been proposed in the literatures that aim to improve the accuracies of the derived effort estimates, and each proposed model is compared with previously defined techniques to see which one is better on a given project or set of projects [15,24]. However, most software projects have incomplete information and uncertain relations between effort drivers and the required development effort.

The grey system theory is a useful method that is applicable to unascertained problems with too little data or information [10,31]. The grey relational analysis (GRA) in the grey system theory is a famous technique that has been used in the last three decades [6,9,17]. A software effort estimation model using the GRA identifies one or more historical projects that are similar to the project that is to be estimated and derives an estimate from them. Users may be more willing to accept solutions coming from the GRA since they are derived from a form of reasoning that is more akin to human problem-solving, as opposed to the somewhat arcane chains of neural nets. This advantage is particularly important when systems are not only to be deployed but are also to be analyzed to determine the reasoning processes behind them.

When designing a software effort estimation model using the GRA, a model builder should identify the effort drivers of a software project that are believed to be significant when determining the relations of software projects. Essentially, the weight in the GRA for an effort driver needs to be increased when such an effort driver is significant in determining the similarity between a pair of projects. In this regard, a good strategy to specify significant effort drivers is to build a learning mechanism whose algorithm can derive the optimal degree of importance for each of the various effort drivers.

Genetic algorithm (GA) is a searching technique that is based on the natural evolution of a species and has been used for solving optimization problems in many fields [14,23]. Thus, GA is used in the GRA learning processes to derive suitable weights for each software effort driver to determine the similarity measures between pairs of projects. The GRA method does not appear to have been used in software effort estimation problem, although it has been proposed and is widely used in many areas [5,6,17,20]. This paper presents the procedure of building a formal software effort estimation using the integration of the GRA with GA, comparing its accuracy with other models using three well-known model-building methods: case-based reasoning, classification and regression trees, and artificial neural networks, in the literature of the software project estimation field.

## 2. Literature review

The grey system theory initiated by Deng is mainly utilized to study uncertainties in system models, analyze relations between systems, establish models, and make forecasts and decisions [9,11,18,27]. The meaning of "grey" can be expressed as the characteristic between black and white. "Black" means that the required information is not entirely available. Conversely, "white" means that the required information is complete. The purpose of the "grey" system and its applications is to bridge the gap between "black" and "white." Incomplete information is the basic characteristic of a grey system. It emphasizes discovering the true properties of systems under poorly-informed situations. The grey system theory seeks only the intrinsic structure of the system given such limited data. The grey system focuses keenly on what partial or limited information the system can provide, and tries to paint its total picture from this.

The GRA is an important method in the grey system theory. Recently, the GRA has become very popular in a number of areas such as manufacturing [20], transportation [17], and the building trade [7]. In the grey system theory, the GRA is essentially believed to have captured the similarity measurements or relations in a system [10]. With a given reference sequence and a given set of comparative sequences, the GRA can be used to determine the grey relational grade (GRG) between the reference and each element in the given set. The best comparative one can be found by analyzing the resultant GRGs. In other words, the GRA can be viewed as a measure of similarity for finite sequences.

The GRA can also be used as a measure of the absolute point-to-point distance between sequences [13]. However, it should be used carefully in practice

for the tendency prediction problem, as it is likely to retrieve the sequence with smaller point-to-point distance between two sequences than the one with the highly similar tendency [4]. For software effort estimation problem, it is not a tendency prediction or a time series problem because the approximation measure is more important than the tendency between two software projects. One of the widely-used software effort estimation methods is the case-based reasoning (CBR) which also utilizes the same concept of measuring point-to-point distance for determining the most similar project for software effort estimation [8,19]. The utilization of the GRA for software effort estimation can be considered as a potential approach in comparison with the CBR.

Software effort estimation using the GRA is a form of CBR estimation model. The GRG between pairs of projects (sequences) plays an important role in the GRA estimation model [6]. The distance measured in the GRA shows the degree of similarity between the two projects in terms of their effort drivers. The retrieval of the appropriate project relies on a relational measure that takes into account the GRG between pairs of projects. An estimation system using the GRA may perform poorly in retrieving projects when effort drivers are irrelevant in the matching projects. It is crucial to identify the most salient effort drivers that leads to the effective project retrieval to minimize the bias associated with the effort drivers. Generally, the performance of the relational measures and the weights of the weighted GRG for each effort driver are key to this reasoning process. Appropriately setting effort driver weights in the weighted GRG can improve the performance of its derived estimates.

Essentially, more important effort drivers should be assigned larger weights for the weighted GRG compared to less important effort drivers. However, most of the GRA applied the GRG with the same weight and did not use significant weight for each factor [7,20]. Some studies improved this issue by deriving weights based on the experience of experts in the GRA [5], but the determination of the effort driver weights highly depended on their subjective opinions. The problem is mainly due to the difficulty in determining the appropriate weights to the complexity factors. How to determine the effort driver weights is a large issue because it is difficult to give suitable values. GA is a stochastic algorithm that uses an adaptive search method for solving problems. It is very useful in optimizing complex search problems, especially in investigating suitable weights.

Holland first introduced GA as an adaptive search technique that mimics the processes of evolution to solve optimization problems [16]. This kind of algorithm is based on natural evolution and on the Darwinian theory of natural selection [14]. GA uses three basic generic operators: selection, crossover and mutation [25]. Selection is a process in which individual solutions are copied according to their fitness values. Goldberg presented a roulette wheel strategy, which is a classical selection operator for general GA [14]. Each member of the pool is assigned a space on a roulette wheel proportional to its fitness. The fittest members would then have the highest probability of selection. Crossover requires a mating of two randomly selected strings. The information on the strings is partly interchanged with a randomly chosen crossover site. Crossover is applied to take valuable information from the parents, and it has a certain probability involved with it. Mutation is an occasional, random alternation of the value of a solution position. It insures against bit loss, and can be a source of new bits as well. Since mutation is a random walk through the solution space, it must be used sparingly.

Most of the existing GRA methods applied the same amount of weight to the GRG. Although some research works dealing with the weighted GRG by experts exist in the literatures, the determination of weights highly relies on subjective judgment. Meanwhile, hardly any research work has adopted the GA approach in deciding the suitable amount of weights in the weighted GRG for the GRA. Thus, this paper aims to investigate the effects of accuracy of the software effort estimation model by integrating the GRA with the GA method.

## 3. Research method

A framework of the formal software estimation model that integrates the GRA with GA is shown in Fig. 1. In the model, the effort drivers are first normalized in the range between 0 and 1, which is also called the grey relational generating. The grey relational coefficient is then calculated from normalized effort drivers to express the relationship between the project to be estimated and the historical projects for each effort driver. After, the weighted GRG is computed by summing up the grey relational coefficients

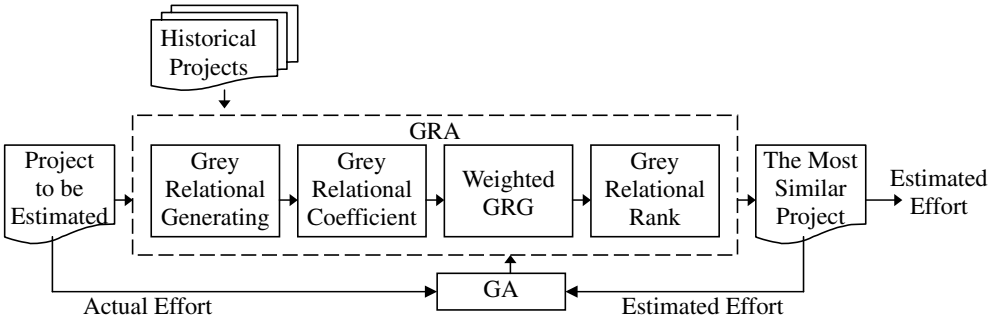


Fig. 1. A framework of software effort estimation by integrating the GRA with GA.

corresponding with each effort driver and their respective weight. The overall evaluation of the multiple effort drivers is based on the weighted GRG. The most similar project is the one with the highest weighted GRG. A higher weighted GRG indicates that the corresponding historical project is closer to the project that is to be estimated. The effort of the historical project that is most similar to the project to be estimated is then retrieved as the estimated effort.

For the weighted GRG, GA is then adapted to search for the suitable weights in the GRA. The weights of the weighted GRG for each effort driver are treated as variables to be searched, and the mean magnitude relative error (*MMRE*) between the actual efforts and the estimated efforts are considered as the value that is to be optimized. This process is based on the principle of "survival of the fittest." The weights of the weighted GRG for each effort driver are encoded in a chromosome, and the collections of chromosomes make up a population. The GA can be thought of as an evolutionary process, where a population of chromosomes evolves over a sequence of generations. In each generation, the fitness of each chromosome is calculated, and chromosomes are selected for reproduction on the basis of their fitness. A chromosome's probability of survival of is proportional to its fitness value. The reproduced chromosomes then undergo a recombination that consists of crossovers and mutations. These iterative processes optimize the weights of the weighted GRG for each effort driver.

### 3.1. Grey relational generating

Grey relational generating is used to replenish messages from the data by finding regularities and properties in jumbled effort drivers. It is also aware of procedures like data pre-processing. Among $n$ projects, a project (an original sequence) with $m$ effort drivers for the $i$th project is shown in Eq. (1). The sets of effort drivers must be standardized because of the different feature values of the various effort drivers. If the range of feature values for one effort driver is much larger than the others, the effect of this effort driver in the GRA is easily neglected. Therefore, the normalization of all effort drivers should be done first in order to overcome this bias

$$x_i^{(0)} = \{x_i^{(0)}(k) | k = 1, 2, \ldots, m\}, \tag{1}$$

where $i \in \{1, 2, \ldots, n\}$.

As the physical meanings of the causal effort drivers in a project could vary, effort drivers with numeric types are transferred by a common transfer rule of "the larger is the best". A grey relational generating formula of effort driver $k$ for the $i$th project is shown in Eq. (2), where $x_i^{(0)}(k)$ means the feature value of effort driver $k$. Among $n$ projects, max $x_i^{(0)}(k)$ represents the maximum value for effort driver $k$ and min $x_i^{(0)}(k)$ reveals the minimum value for effort driver $k$

$$x_i(k) = \frac{x_i^{(0)}(k) - \min x_i^{(0)}(k)}{\max x_i^{(0)}(k) - \min x_i^{(0)}(k)}, \tag{2}$$

where $k \in \{1, 2, \ldots, m\}, i \in \{1, 2, \ldots, n\}$.

The grey relational generating is applied when transferring the original projects $x_i^{(0)}$ to a set of comparable projects as denoted by $x_i$. Among $n$ projects, the project with $m$ effort drivers for the $i$th project is shown in Eq. (3), where $i \in \{1, 2, \ldots, n\}$ and $k \in \{1, 2, \ldots, m\}$

$$x_i = \{x_i(k) | k = 1, 2, \ldots, m\}, \tag{3}$$

where $i \in \{1, 2, \ldots, n\}$.

### 3.2. Grey relational coefficient

The grey relational coefficient is calculated to express the relation between the project to be estimated (reference sequence) and historical projects (sequences to be compared) for each effort driver. Supposing the project to be estimated is $x_0 = \{x_0(1), x_0(2), \ldots, x_0(k)\}$, where $k \in \{1,2,\ldots,m\}$, and the historical projects are $x_i = \{x_i(1), x_i(2), \ldots, x_i(k)\}$, and where $i \in \{1,2,\ldots,n\}$. The formula then of the grey relational coefficient of project $x_i$–$x_0$ at the $k$th effort driver is given in Eq. (4). The grey relational coefficient can be intuitively considered as the point-to-point relation at the $k$th effort driver between $x_0$ and $x_i$. Here, $\Delta_{0i}(k)$ stands for the difference between $x_0$ and $x_i$ (Norm). The distinguishing coefficient $\zeta$ is included to adjust the difference between $\Delta_{0i}(k)$ and $\Delta_{\max}$

$$\gamma(x_0(k), x_i(k)) = \frac{\Delta_{\min} + \zeta\Delta_{\max}}{\Delta_{0i}(k) + \zeta\Delta_{\max}}, \tag{4}$$

where $\Delta_{0i}(k) = |x_0(k) - x_i(k)|$, $\Delta_{\min} = \min_{\forall i}\min_{\forall k} \Delta_{0i}(k)$, $\Delta_{\max} = \max_{\forall i}\max_{\forall k}\Delta_{0i}(k)$, $\zeta \in (0,1]$.

### 3.3. Grey relational grade

The GRG is a globalized measure adopted for the GRA. It is used to describe and explain the relation between two sets under a certain background. The degree of similarity between them is easily determined using the GRG. The greater GRG between two projects, the closer the relationship between these projects are. In other words, the more alike the two projects are, the greater of GRG is.

For software effort estimates, GRG can present the degree of the relationship between the project to be estimated and its historical projects. The GRG is the mean of grey relational coefficient at each effort driver. The range of the GRG is a closed interval between 0 and 1. In reality, the GRA compares the relationship of the projects in their appropriate metric spaces. If two projects agree at all effort drivers, then their grey relational coefficient is 1 in everything, and therefore their GRG should also be 1. In view of this, the GRG of two comparing projects can be quantified by the mean value of their grey relational coefficients as shown in Eq. (5). Here, $r(x_0, x_i)$ is designated as the GRG between $x_i$ and $x_0$, and $m$ is the number of effort drivers

$$r(x_0, x_i) = \frac{1}{m}\sum_{k=1}^{m} r(x_0(k), x_i(k)), \tag{5}$$

where $i \in \{1,2,\ldots,n\}$.

Considering the contribution from different effort drivers to the GRA in software effort estimates, the mean of the grey relational coefficient between $x_i$ and $x_0$ is shown in Eq. (6). Among $m$ effort drivers, $W_k$ is the weight for effort driver $k$. Each effort driver has its respective weight in conjunction with its grey relational coefficient for the weighted GRG. If all effort drivers have the same weight $1/m$, Eq. (6) is then equal to Eq. (5)

$$r(x_0, x_i) = \sum_{k=1}^{m} W_k * r(x_0(k), x_i(k)), \tag{6}$$

where $i \in \{1,2,\ldots,n\}$, $\sum_{k=1}^{n} W_k = 1$.

The weights in the weighted GRG indicate the significance of effort drivers in determining the similarity between two projects. The GA is applied to search for the suitable weights in conjunction with the grey relational coefficients for the weighted GRG in the GRA. An evolution process of GA for exploring weights is shown in Fig. 2. The weights of weighted GRG are treated as variables to be searched, and the candidate solution of the weights is represented as a string called a chromosome. Some chromosomes (i.e. solutions) make up the initial population that is composed by randomly selecting the initial values for the weights. Fitness evaluation measures the *MMRE* between the actual and estimated efforts for each project being estimated. The objective of GA is to find the suitable weights of effort drivers which produce the minimum *MMRE* among a set of projects being

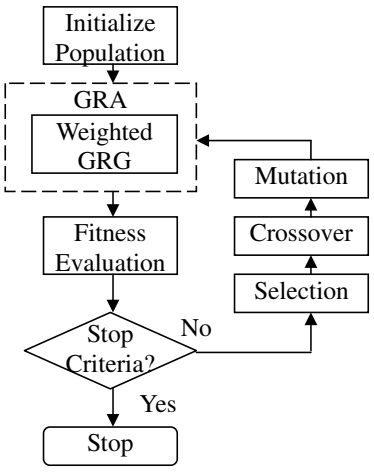


Fig. 2. The evolution process of GA for exploring weights.

estimated. The stop criteria determine whether the present weights need to be further explored or not.

Three basic generic operators of selection, crossover and mutation produce a new generation of chromosome that is potentially more suitable than the previous solutions. Selection is a process in which individual chromosomes are duplicated according to their fitness values; while crossover is partly interchanged with two randomly selected chromosomes with a certain probability. Mutation operation follows the crossover operation and determines if a solution or chromosome should be mutated in the next generation. GA uses these three basic generic operators and evaluates goodness of the chromosome on each generation to produce better optimal weights for the weighted GRG.

### 3.4. Grey relational rank

For the project being estimated, the estimated effort should be retrieved from the historical project with the largest weighted GRG among all historical projects. On the basis of the weighted GRGs between the project to be estimated and the historical projects, we can rank all the projects in accordance to their weighted GRGs. This procedure is called the grey relational rank. While the practical meaning of the numerical values of the weighted GRGs among these projects is not absolutely important, the grey relational rank between them, on the other hand, yields more subtle information.

The weighted GRG, $r(x_0, x_i)$ represents up to what degree of inference the project $x_i$ can exert influence on the project to be estimated. We can evaluate the weighted GRGs between the project $x_0$ and each of the historical project $x_a$ or $x_b$. If $r(x_0, x_a) > r(x_0, x_b)$, we say $x_a$ to $x_0$ is better than $x_b$ to $x_0$. If $r(x_0, x_a) < r(x_0, x_b)$, we say $x_a$ to $x_0$ is worse than $x_b$ to $x_0$. Otherwise, if $r(x_0, x_a) = r(x_0, x_b)$, we say $x_a$ to $x_0$ is of equal worth to $x_b$ to $x_0$. The historical project that is the most similar to the project to be estimated is consequently confirmed, as shown in Eq. (7). $\hat{x}_0(Y)$ represents the estimated effort for project $x_0$, and $x_a(Y)$ as the actual effort for project $x_a$. The estimated effort for project $x_0$ is therefore determined by the actual effort of the historical project $x_a$, with the maximum weighted GRG included among all historical projects

$$\text{if } r(x_0, x_a) = \max\{r(x_0, x_i)\} \text{ Then } \hat{x}_0(Y) = x_a(Y), \tag{7}$$

where $i \in \{1, 2, \ldots, n\}$, $a \in i$, $Y \notin k$.

### 3.5. Evaluation criteria

The degree to which the model's estimated effort matches the actual effort is one of the principles in evaluating a software effort estimation model. A fundamental question that needs to be asked in all estimation methods is how accurate the estimates are. The most widely accepted criterion is the mean magnitude relative error (*MMRE*). Generally, an acceptable target value for *MMRE* is 25%, which indicates that, on the average, the accuracy of the established estimation models should be less than 25%. The formula for *MMRE* is based on the calculations of the magnitude relative error (*MRE*). *MRE* calculates each project in a data set while the *MMRE* aggregates the multiple projects. Eq. (8) shows a *MRE* criterion that is used to assess the efficiency of an effort estimate for project $x_i$, where $x_i(Y)$ represents the actual effort and $\hat{x}_i(Y)$ as the estimated effort. Among $n$ projects, the estimating accuracy of *MMRE* is the mean of all *MREs*, as shown in Eq. (9)

$$MRE_i = \frac{|x_i(Y) - \hat{x}_i(Y)|}{x_i(Y)}, \tag{8}$$

$$MMRE = \frac{\sum_{i=1}^{n} MRE_i}{n}. \tag{9}$$

Another widely-used criterion which presents the percentage of estimates that fall within $l$ percent of the actual value is *Pred*($l$). This measure shows a proportion of the projects in a given level of accuracy and is often used in the literatures. The definition of *Pred*($l$) is shown in Eq. (10), where $n$ is the total number of observations and $K$ is the number of observations whose *MRE* is less than or equal to $l$. A common value for $l$ is 0.25, which is also used in the present study. The *Pred*(0.25) represents the percentage of projects whose *MRE* is less than or equal to 25%. We adopt the *MMRE* and *Pred*(0.25) as prediction performance indicators in this paper since they are widely used in the literatures, and thereby render our results to be more comparable with other works

$$Pred(l) = \frac{K}{n}. \tag{10}$$

## 4. Experiment results

The data used in this study came from two sources: the COCOMO database [2] and the Albrecht data set [1]. All numeric effort drivers were

transferred to the comparable space in order for them to have the same degree of influence. These effort drivers were normalized based on "the larger is the best" principle in both the COCOMO database and the Albrecht data set. We examined our experiment results using a threefold cross-validation approach [3] because it yields more realistic accuracy measures.

The above two data sets were used to establish the proposed software effort estimation model by integrating the GRA with GA. Their performances were also investigated. For the parameter of the GRA model, a middle value of 0.5 for the grey distinguishing coefficient was used. Regarding the GA parameters, 50 organisms in the population were used. The crossover rate varied from 0.5 to 0.6, whereas the mutation rate ranged from 0.06 to 0.1 in the initial settings. The entire learning process stopped after 5000 trials, when the best result did not change in the last 500 trials. The evaluation criterion of the *MMRE* was treated as its fitness function.

We utilized *MMRE*, *Pred*(0.25) and boxplots to compare the training and testing performances of the proposed GRA model with three famous software effort estimation models of case-based reasoning (CBR) [26], classification and regression trees (CART) [30], and artificial neural networks (ANN) [15].

### 4.1. COCOMO data set

Sixty-three historical projects in the COCOMO data set were adopted as an example in the construction of the GRA software effort estimation model. There were 17 effort drivers in conjunction with an actual effort for each project in this data set. The actual effort in the COCOMO data set was given in units of person-month (PM). PM is the number of months that one person would need to develop a given project. The distribution of the effort in this data set is displayed in hundred PM units, as shown in Fig. 3. The effort did not result in uniform distribution or normal distribution. Most of the data is confined to <1000 PM range, and very little data is available to numbers above 10,000 PM. It is, in any case, unrealistic to demand a uniformly distributed or normally distributed historical data set. This data thus provides a challenging learning problem to a software effort estimation model using GRA and provides a good test for generalized power obtained through various learning algorithms.

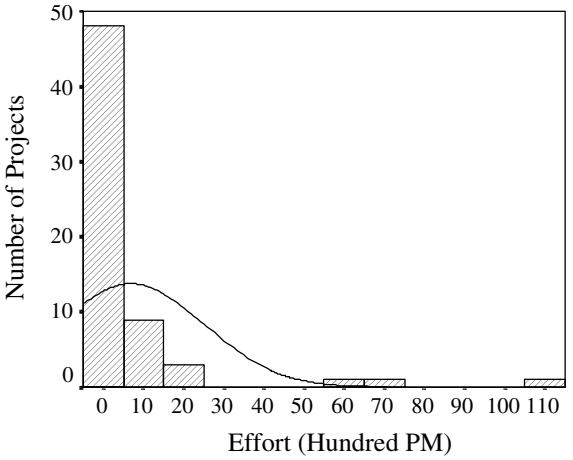


Fig. 3. Effort distribution of the COCOMO data set.

Table 1 shows the estimated results at both the training and testing stages of the four effort estimation models based on the *MMRE* and *Pred*(0.25) evaluation criteria. A software effort estimation model with lower *MMRE* value shows that its derived estimates have better accuracy than the others. Regarding the *MMRE* evaluation criterion for each model, the GRA model outperforms the three software effort estimation models: the CBR, CART and ANN at both training and testing phases. Furthermore, an estimation model with higher *Pred*(0.25) values presents its estimates as generally more accurate than the others for a given level of accuracy at 0.25. The testing result of the GRA model also has the highest *Pred*(0.25) compared to the other three models of the CBR, CART and ANN. The GRA model resulted in almost consistent estimates in both *MMRE* and *Pred*(0.25) and outperformed the other three software effort estimation models.

A diagrammatic explanation of the *MRE* for each project allows visual comparison of different

Table 1
Estimation results of the COCOMO data set

| Models | *MMRE* | | *Pred*(0.25) | |
|---|---|---|---|---|
| | Training | Testing | Training | Testing |
| GRA | 0.53 | 0.69 | 0.41 | 0.38 |
| CBR | 2.19 | 4.46 | 0.09 | 0.12 |
| CART | 1.22 | 2.44 | 0.44 | 0.25 |
| ANN | 1.41 | 1.43 | 0.11 | 0.11 |

models. As shown in Fig. 4, the boxplots illustrate the median, the interquartile range, and the outlier projects of individual models. The box represents the interquartile range that contains 50% of the projects. A line across the box indicates the median value of *MRE*, and the circle represents the outlier project. A wide range of upper and lower tails reveals a large distribution of the *MRE* values. Among these models, the CBR and CART models have a large distribution of the *MRE* values. The estimation results of the GRA model are almost unbiased and have more consistent estimates compared to other models. These encouraging results show that this method of integrating GRA with GA in order to build a formal software effort estimation model is more effective than the CBR, CART and ANN methods.

A study made by Dolado provided the estimation results from the COCOMO data set based on its regression analysis and genetic programming methods [12]. The model of the regression analysis resulted in $Pred(0.25) = 0.17$ and $MMRE = 1.13$. The genetic programming model resulted in $Pred(0.25) = 0.15$ and $MMRE = 1.78$. By comparing both criteria of the *MMRE* and *Pred*(0.25), the GRA method also outperforms the regression analysis and genetic programming methods. Thus, the effectiveness of the proposed GRA software effort estimation model is clearly demonstrated in the COCOMO data set.

### 4.2. Albrecht data set

The Albrecht data set came from the IBM data processing services database, which includes projects with medium to large sizes. This database contained business applications that were developed using third-generation languages. Five effort drivers – input count, output count, query count, file count and adjust factors – were considered for the model construction. One project in the data set had an effort that was more than twice as small as the second smallest project. In practice, this project would not be suitable to be an analogue for other projects. Hence, this project was regarded as outlying and was thus excluded from the data set. This process resulted in 23 projects that could be used for our study. Fig. 5 displays the effort distribution of project data in the Albrecht data set after excluding the outlying project. The effort in this data set is recorded in thousand hours increments.

On the four software effort estimation models, the performance indicators of the *MMRE* and *Pred*(0.25) at both training and testing stages on the Albrecht data set are shown in Table 2. On the *MMRE* results, the GRA model is shown to be superior to the other models of CBR, CART and ANN at both training and testing stages. Hence, the GRA software estimation model has

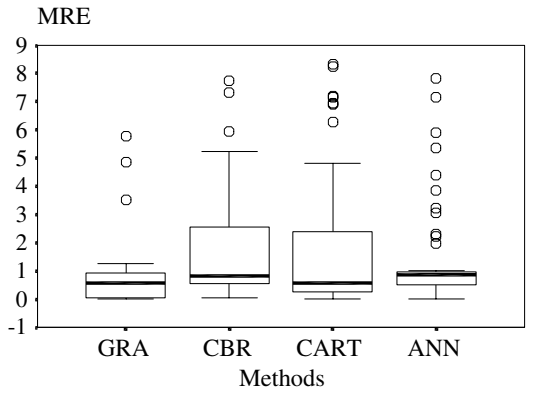


Fig. 4. Boxplots of *MRE* in the COCOMO data set.

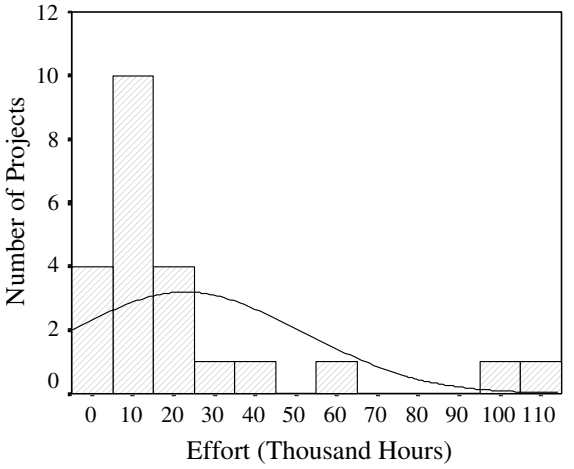


Fig. 5. Effort distribution of the Albrecht data set.

Table 2
Estimation results of the Albrecht data set

| Models | *MMRE* | | *Pred*(0.25) | |
|---|---|---|---|---|
| | Training | Testing | Training | Testing |
| GRA | 0.37 | 0.31 | 0.49 | 0.48 |
| CBR | 0.58 | 0.58 | 0.32 | 0.39 |
| CART | 0.71 | 0.76 | 0.36 | 0.26 |
| ANN | 0.79 | 0.86 | 0.30 | 0.21 |

been proven to significantly improve the estimation accuracy. In addition, the GRA model has the highest *Pred*(0.25) value compared to the other models. The results of the GRA model present consistent performances in both the *MMRE* and the *Pred*(0.25) evaluation criteria. Thus, the effectiveness of the proposed software effort estimation by integrating the GRA with GA is also clearly demonstrated in the Albrecht data set as well.

Fig. 6 depicts the prediction results measured by the *MRE* using boxplots. The height of the box, which represents the interquartile range, is larger for the CART and ANN models. This indicates a high dispersion of the *MRE* values for these two models compared to the other software effort estimation models of the CBR and GRA. Furthermore, the ANN model shows the largest upper and lower end of the *MRE* values compared to the other three GRA, CBR and CART models. The distribution of the *MRE* observation for the ANN model is the largest compared to the others. When comparing the GRA model with other estimation models, the GRA model possessed the smallest interquartile range, which indicates that the estimates of the GRA model are unbiased and are consistently compared to the other three software effort estimation models.

Shepperd and Schofield compared the estimation accuracies of the regression analysis and analogy software estimation models by using the Albrecht data set [28]. Their experimental results showed that the regression analysis model resulted in $Pred(0.25) = 0.33$ and $MMRE = 0.90$, and the analogy model $Pred(.25) = 0.33$ and $MMRE = 0.62$. The proposed GRA software effort estimation model in this paper also outperforms the above regression analysis and analogy model results in

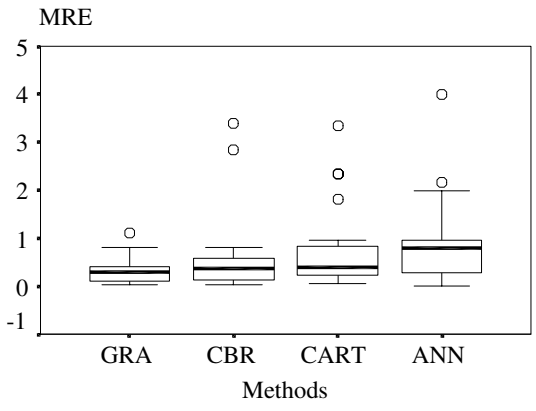


Fig. 6. Boxplots of *MRE* on the Albrecht data set.

both the *MMRE* and *Pred*(0.25). These results show that the software effort estimation model using the integration of the GRA with GA also presents a better prediction accuracy compared to the other three software estimation models on the Albrecht data set.

### 4.3. Improvements of applying GA to GRA

The improvement of the GRA with and without GA models compared with CBR, CART and ANN methods in both the COCOMO and Albrecht data sets at the testing stage is shown in Table 3. For the COCOMO data set, the improvement of the GRA without GA model is between −85.3% and 81.8% in comparison with CBR, CART and ANN models. It shows that the GRA without GA model presents insignificant difference compared with those methods often used by former researchers. However, the improvement of the GRA with GA model in comparison with CBR, CART and ANN models is between 51.7% and 245.5%. This demonstrates that the GRA with GA model outperforms the GRA without GA model in the COCOMO data set.

For the Albrecht data set, the improvement of the GRA without GA model is between 12.8% and 109.5% in comparison with CBR, CART and ANN models. Likewise, the improvement of the GRA with GA model in comparison with CBR, CART and ANN models is between 23.1% and 128.6%. Although the GRA without GA model presents slight improvement compared with CBR, CART and ANN models, it still demonstrates that the GRA with GA model is superior to the GRA without GA model and the CBR, CART and ANN models often used by former researchers in the Albrecht data set.

Integration of the GRA with GA model for software effort estimation is the main innovation of this work. To show the improvement caused by this integration, Table 4 shows the improvement of *MMRE* and *Pred*(0.25) evaluation criteria when applying GA to the GRA model for optimizing weights of effort drivers at both the training and testing stages. For the *MMRE* evaluation criterion, the highest improvement is 76.2% at the training stage and 74% at the testing stage in the COCOMO and Albrecht data sets. The mean improvement for these two data sets is 54.5% at the training stage and 39.9% at the testing stage. For the *Pred*(0.25) evaluation criterion, the highest improvement is 355.6% at the

Table 3
Improvements with different models

| Data set | Approach | CBR | | CART | | ANN | |
|---|---|---|---|---|---|---|---|
| | | *MMRE* (%) | *Pred*(0.25) (%) | *MMRE* (%) | *Pred*(0.25) (%) | *MMRE* (%) | *Pred*(0.25) (%) |
| COCOMO | GRA without GA | 40.6 | 66.7 | −8.6 | −20 | −85.3 | 81.8 |
| | GRA with GA | 84.5 | 216.7 | 71.7 | 52 | 51.7 | 245.5 |
| Albrecht | GRA without GA | 41.4 | 12.8 | 55.3 | 69.2 | 60.5 | 109.5 |
| | GRA with GA | 46.6 | 23.1 | 59.2 | 84.6 | 64 | 128.6 |

Table 4
Improvement of applying GA to GRA

| Data set | Approach | *MMRE* | | *Pred*(0.25) | |
|---|---|---|---|---|---|
| | | Training | Testing | Training | Testing |
| COCOMO | GRA without GA | 2.23 | 2.65 | 0.09 | 0.2 |
| | GRA with GA | 0.53 | 0.69 | 0.41 | 0.38 |
| | Improvement (%) | 76.2 | 74 | 355.6 | 90 |
| Albrecht | GRA without GA | 0.55 | 0.34 | 0.28 | 0.44 |
| | GRA with GA | 0.37 | 0.32 | 0.5 | 0.48 |
| | Improvement (%) | 32.7 | 5.9 | 78.6 | 9.1 |
| Mean improvement (%) | | 54.5 | 39.9 | 217.1 | 49.5 |

training stage and 90% at the testing stage in these two data sets. The mean improvement of *Pred*(0.25) evaluation criterion is 217.1% at the training stage and 49.5% at the testing stage in these two data sets.

In general, the estimation accuracy of the GRA without GA model for software effort estimation is not significantly improved based on the *MMRE* and *Pred*(0.25) evaluation criteria in the COCOMO and Albrecht data sets compared with CART, ANN and CBR models. However, the estimation accuracy is significantly improved when applying GA to the GRA model for optimizing weights of effort drivers compared with CART, ANN and CBR methods often used by former researchers.

## 5. Conclusions

Delivering reliable and accurate estimates of software development effort has always been a challenge for both the software industry and academic communities. An expert-based software effort estimation model is the commonly used method in current software engineering practices, but the derived estimates are subjective and are thus often unacceptable. Software effort estimation using the GRA is a technique that is able to search for similarities from historical projects and can accordingly derive an estimate from the identified similar project. Most GRA models utilize the GRG with the same weight approach to identify other similar projects. Finding the proper weight for each of the software effort driver in GRA model is difficult, but it is a key factor to enhance the prediction accuracy of the GRA software effort estimation model.

This paper has investigated the impacts of integrating the GRA with GA in deciding the appropriate weights of the weighted GRG for each effort driver to improve the accuracies of the software effort estimates. Experimental results were encouraging when GA was applied to the GRA to improve the accuracies of software effort estimates in both the COCOMO database and the Albrecht data set. The integration of the GRA with GA method has generated better software effort estimates from the results of the *MMRE* and *Pred*(0.25) compared to other popular software effort estimation methods in the literatures. Hence, the experimental result demonstrated that the integration of the GRA with GA is a feasible approach that can supply the objective weights to the weighted GRG for each software effort driver. It also revealed that the GRA effort estimation model with weighted GRG presents a better predicting accuracy over the results using case-based reasoning, classification and regression trees, as well as artificial neural networks methods.

The integration of the GRA with GA method for software effort estimation accounts for the extra complexity in the model construction. This may affect the use of the proposed method for the software effort estimation in practice. As an automatic

software tool can largely reduce the extra complexity, the use of an evolutionary model, such as this proposed method in this paper, for software effort estimation is worth the gains. It is difficult to propose a software effort estimation model that can provide both the accurate estimates and the simple mechanism in the model construction. Hence, future works can further evaluate different approaches to obtain the feature weights for GRA, which are able to provide more accurate estimates and the simple model-construction mechanism for software effort estimation.

The performance of the proposed GRA software effort estimation model in this paper is largely dependent on the historical software project data itself. This implies that the data collected should be as precise as possible. Essentially, the collection of more diverse software projects is required to generate more accurate software effort estimates. Despite the restriction of the project data set, the integration of the GRA with GA has shown its ability to generate adequate effort estimates. We are encouraged by the results of this present study and are interested in investigating whether further improvements can be made by another design of the GRA software effort estimation model. Meanwhile, the proposed approach in this paper can also be applied to different software project estimation domains such as software size and software quality estimation models.

## Acknowledgements

This research was supported by the National Science Council (NSC) of Taiwan under the contract 94-2416-H-011-001. The authors also wish to thank anonymous reviewers for their constructive comments and the Editor Dr. L. Peccati for his editorial effort on the manuscript of this paper.

## References

[1] S.J. Albrecht, J.R. Gaffney, Software function, source lines of code, and development effort prediction: A software science validation, IEEE Transactions on Software Engineering 9 (6) (1983) 639–648.
[2] B. Boehm, Software Engineering Economics, Prentice-Hall, Englewood Cliffs, NJ, 1981.
[3] P. Burman, A comparative study of ordinary cross-validation, V-fold cross-validation and the repeated learning-testing methods 76 (1989) 503–514.
[4] B.R. Chang, Novel grey relational measurements, International Joint Conference on Neural Networks 3 (2001) 1615–1619.
[5] J.R. Chang, C.T. Hung, G.H. Tzeng, J.D. Lin, Non-additive grey relational model: Case study on evaluation of flexible pavement, in: Proceedings of the IEEE International Conference on Fuzzy Systems, 2004, pp. 577–582.
[6] K.C. Chang, M.F. Yeh, Grey relational analysis based approach for data clustering, IEEE Proceedings-Visual Image Signal Process 152 (2) (2005) 165–172.
[7] W.C. Chang, K.L. Wen, H.S. Chen, T.C. Chang, The selection model of pavement material via grey relational grade, in: Proceeding of the IEEE International Conference on Systems, Man, and Cybernetics, 2000, pp. 3388–3391.
[8] N.H. Chiu, S.J. Huang, The adjusted analogy-based software effort estimation based on similarity distances, Journal of Systems and Software 80 (2007) 628–640.
[9] J. Deng, Control problem of grey systems, Systems and Control Letters 1 (5) (1982).
[10] J. Deng, Grey information space, The Journal of Grey System 1 (1989) 103–117.
[11] J. Deng, Introduction to grey system theory, The Journal of Grey System 1 (1989) 1–24.
[12] J.J. Dolado, On the problem of the software cost function, Information and Software Technology 43 (2001) 61–72.
[13] C.P. Fung, Manufacturing process optimization for wear property of fiber–reinforced polybutylene terephthalate composites with grey relational analysis, Wear 254 (2003) 298–306.
[14] D. Goldberg, Genetic Algorithms in Search Optimization and Machine Learning, Addison-Wesley Publishing, 1989.
[15] A. Heiat, Comparison of artificial neural network and regression models for estimating software development effort, Information and Software Technology 44 (2002) 911–922.
[16] J. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, USA, 1975.
[17] Y.T. Hsu, C.B. Lin, S.F. Su, High noise vehicle plate recognition using grey system, The Journal of Grey System 10 (1998) 193–208.
[18] C.I. Hsu, Y.H. Wen, Application of grey theory and multiobjective programming towards airline network design, European Journal of Operational Research 127 (1) (2000) 44–68.
[19] S.J. Huang, N.H. Chiu, Optimization of analogy weights by genetic algorithm for software effort estimation, Information and Software Technology 48 (2006) 1034–1045.
[20] B.C. Jiang, S.L. Tasi, C.C. Wang, Machine vision-based gray relational theory applied to IC marking inspection, IEEE Transactions on Semiconductor Manufacturing 15 (4) (2002) 531–539.
[21] M. Jorgenson, A review of studies on expert estimation of software development effort, The Journal of Systems and Software 70 (2004) 37–60.
[22] T. Jorgensen, S.W. Wallace, Improving project cost estimation by taking into account managerial flexibility, European Journal of Operational Research 127 (2) (2000) 239–251.
[23] J.J. Kelly, L. Davis, A hybrid genetic algorithm for classification, in: Proceedings of the International Joint Conference on Artificial Intelligence, 1991, pp. 645–650.
[24] C.F. Kemerer, An empirical validation of cost estimation models, Communication of ACM 30 (5) (1987) 416–429.
[25] E. Mayer, Genetic Algorithms for Vertex Splitting in DAGS, Department of Computer Science, University of Missouri Rolla, 1993.

[26] T. Mukhopadhyay, S.S. Vicinanza, M.J. Prietula, Examining the feasibility of a case-based reasoning model for software effort estimation, MIS Quarterly 16 (1992) 155–171.
[27] D.K. Ng, Grey system and grey relational model, ACM SIGICE Bulletin 20 (2) (1994).
[28] M. Shepperd, C. Schofield, Estimating software project effort using analogies, IEEE Transactions on Software Engineering 23 (12) (1997) 736–743.
[29] I. Stamelo, L. Angelis, M. Morisio, E. Sakellaris, G.L. Bleris, Estimating the development cost of custom software, Information and Management 40 (2003) 729–741.
[30] E. Stensrud, Alternative approaches to effort prediction of ERP projects, Information and Software Technology 43 (2001) 413–423.
[31] C.C. Wu, N.B. Chang, Gray input–output analysis and its application for environmental cost allocation, European Journal of Operational Research 145 (1) (2003) 175–201.