Available online at www.sciencedirect.com

ScienceDirect

International Journal of Project Management 29 (2011) 56–65

International Journal of Project Management

www.elsevier.com/locate/ijproman

Evolutionary fuzzy decision model for cash flow prediction using time-dependent support vector machines

Min-Yuan Cheng [1], Andreas F.V. Roy *

Dept. of Construction Engineering, National Taiwan University of Science and Technology, #43, Sec. 4, Keelung Rd., Taipei, Taiwan, ROC

Received 17 September 2009; received in revised form 25 December 2009; accepted 19 January 2010

Abstract

The ability of project managers to make reliable cash flow predictions enhances project cost flow control and management. Reliable cash flow prediction over the course of a construction project puts the project manager in a better position to identify potential problems and develop appropriate strategies to mitigate the negative effects of such on overall project success. Therefore, managers should monitor project progress using cash flow data, which has unique characteristics, as time series data. However, the complex, mutable nature of construction projects currently requires significant reliance on experience and expert opinions to predict cash flow on an ongoing basis. Recent studies have indicated good potential for using artificial intelligence to reduce reliance on human input in cash flow prediction processes. The Evolutionary Fuzzy Support Vector Machine Inference Model for Time Series Data (EFSIM$_T$), an artificial intelligence hybrid system focusing on the management of time series data characteristics which fuses fuzzy logic (FL), weighted support vector machines (weighted SVMs) and a fast messy genetic algorithm (fmGA), represents a promising alternative approach to predicting cash flow. Simulations performed on historical cash flow data demonstrate the EFSIM$_T$ is an effective tool for predicting cash flow.
© 2010 Elsevier Ltd and IPMA. All rights reserved.

Keywords: Cash flow prediction; Time series; Fuzzy Logic; Weighted Support Vector Machine; Fast Messy Genetic Algorithms

1. Introduction

Compared to other businesses, the construction industry faces higher risks due to significant uncertainties inherent in the operating environment. A considerable proportion of business failures in this sector can be attributed to financial factors (Touran et al., 2004; Khosrowshahi and Kaka, 2007). Russell (1991) stated that the primary instigator in over 60% of construction company failures is financial trouble, attributable to economic factors such as insufficient profits, high interest rates, loss of market, reduced consumer spending and negative or stagnant sector growth.

During project implementation, cash flow is the most critical factor affecting profitability (Hwee and Tiong, 2002). Good operational performance in terms of cash flow control and management impacts significantly upon project management and project success. In light of the vital role played by proper cash flow management, early knowledge of project cash flow trends should represent a critical advantage in terms of ensuring project profitability and creating competitive advantage. Hence, the project manager who continuously monitors cash flow management has a better chance to identify and control potential problems and minimize negative effects that may impact upon overall project success (Ko, 2002; Hwang and Liu, 2005). Russell et al. (1997) opined that such an approach may employ time-dependent variables that change through the construction progress. However, predicting project performance dynamically in terms of cash flow is exceedingly difficult, as each time point is associated numerous time-dependent variables.

Using time-dependent variables during project implementation to proactively control project performance

* Corresponding author. Tel.: +886 2 27301277; fax: +886 2 27301074.
E-mail addresses: myc@mail.ntust.edu.tw (M.-Y. Cheng), d9505803@mail.ntust.edu.tw, roy_afvr@yahoo.com (A.F.V. Roy).

[1] Tel.: +886 2 27336596; fax: +886 2 27301074.

0263-7863/$36.00 © 2010 Elsevier Ltd and IPMA. All rights reserved.
doi:10.1016/j.ijproman.2010.01.004

sequentially in time as the project progresses is a time series problem. Fundamentally, the goal of time series problems is to predict or forecast future values using series history and, potentially, other relevant series or factors (Cryer and Chan, 2008). Various methods and approaches that use traditional statistical methods or AI techniques have been developed to deal with time series problems as well as to forecast and control cash flow. A survey conducted by Sapankevych and Sankar (2009) found time series analysis methods including autoregressive filters, Kalman filters, artificial neural networks (ANNs) and support vector machines (SVMs) have been applied in various fields. They also found that the most important current application of time series analysis is in financial forecasting.

Boussabaine and Kaka (1998) employed a neural network in cash flow forecasting and control in order to overcome the inability of prevalent models based on the regression technique to perform multi-attributes nonlinear mapping. In addition, Boussabaine and Elhag (1999) used a fuzzy technique that increased the effectiveness of cash flow analysis conducted under conditions in which cash flow at particular valuation stages is uncertain. Park et al. (2005) proposed a cash flow forecasting model for construction projects that considered both variable cost weights and time lag. Based on their findings, this model asserted that most previous models developed to predict cash flow addressed realities current in the planning phase, but not necessarily in the construction phase, where uncertain factors can impact upon costs, resource allocations, and timelines.

Currently, artificial intelligence (AI) techniques are considered an alternative approach to solving construction management problems. Some researchers also have been working to combine different AI techniques, as fusing different AI techniques can achieve model performance better than that possible using only one technique (Ko, 2002; Cheng and Wu, 2009). The primary objective of this research study was to facilitate a proactive approach to monitor cash flow management as part of construction project performance control mechanisms by developing an AI hybrid system that fused fuzzy logic (FL), weighted support vector machines/(weighted SVMs) and fast messy genetic algorithms (fmGA). To achieve this objective, FL, weighted SVMs and fmGA were fused into an Evolutionary Fuzzy Support Vector Inference Model for Time Series Data (EFSIM$_T$). EFSIM$_T$ searched simultaneously for fittest distribution of membership functions (MFs) and defuzzification parameters as well as for fittest weighted SVMs hyperparameters and lower bound of weighted data. The summit and width representation method (SWRM) was used to encode MFs (Ko, 2002). The proposed system was verified and validated using data gathered from a construction contractor in Taipei. Data was presented in terms of standard cumulative cost-time curves generated in the process of executing condominium high rise projects between 1996 and 2006. In addition, the performance of the proposed system was compared with SVMs and the Evolutionary Support Vector Inference Model (ESIM) developed by Cheng and Wu (2009).

2. Overview of time series analysis, FL, weighted SVMs and fmGA

2.1. Time series analysis

Time series analysis is a powerful data analysis technique. Fundamentally, the goal of time series analysis is twofold. According to Cryer and Chan (2008) the first goal is to find a suitable mathematical model for data, while the second is to predict or forecast future values in the series based on established patterns and, possibly, other related series or factors.

Over the past several decades, volumes of technical literature have been written about linear prediction in time series analysis, covering such approaches as smoothing methods, the Box and Jenkins time series model and auto regression model. Accurate and unbiased estimation of time series data produced by these linear techniques cannot always be achieved, as real word applications are generally not amenable to linear prediction techniques (Sapankevych and Sankar, 2009). Real world time series applications are fraught by highly nonlinear, complex, dynamic and uncertain conditions in the field. Thus, estimation requires development of a more advanced time series prediction algorithm, such as that developed using an AI approach.

Refenes et al. (1997) expressed that structural change is a time series data characteristic that should always be taken into account in all methodological approaches to time-series analysis. In the light of this characteristic, Cao et al. (2003) expressed that more recent data could provide more relevant information than could distant data. Consequently, recent data should be assigned weights that are relatively greater than weights assigned to earlier data. Cao et al. (2003), Khemchandani et al. (2009) effectively adopted this approach, using AI techniques such as SVMs and weighted SVMs, in financial time series forecasting applications.

2.2. Fuzzy Logic

Fuzzy Logic (FL), a popular AI technique invented by Zadeh in 1960s, has been used in forecasting, decision making and action control in environments characterized by uncertainty, vagueness, presumptions and subjectivity (Bojadziev and Bojadziev, 2007). FL simulates the human decision-making process by employing approximate reasoning logic (Zadeh, 1965). Heshmaty and Kandel (1985) expressed that FL provides a more realistic approach than that used by traditional mathematical models to address phenomena in nature characterized by vagueness and uncertainty.

FL consists of a set of rules that relates a set of inputs to a set of outputs. Quantitative relationships are established through an MF between actual variable values and qualitative, linguistic variables used in ‘if-then’ rules. Therefore, linguistic variables described by MFs and fuzzy if-then rules play an essential role in FL applications (Zadeh, 1973).

FL consists of four major components: fuzzification, rule base, inference engine and defuzzification. Fuzzification is a process that uses MFs to convert the value of input variables into corresponding linguistic variables. The result, which is used by the inference engine, stimulates the human decision-making process based on fuzzy implications and available rules. In the final step, the fuzzy set, as the output of the inference process, is converted into crisp output. This process, which reverses fuzzification, is called defuzzification (Klir and Yuan, 1995).

Despite the advantages of FL, the approach has a number of problems; including identifying appropriate MFs and number of rules for application. This process is subjective in nature and reflects the context in which a problem is viewed. The more complex the problem, the more difficult MF construction and rules become (Ko, 2002). These shortcomings are seen by some researchers as optimization problems, as determining MF configurations and fuzzy rules is complicated and problem oriented. To overcome remaining difficulties, some researchers have tried to fuse FL with AI optimization techniques such as sGA and ant colony (Ishigami et al., 1995; Martinez et al., 2008). These optimization methods have demonstrated their ability to minimize time-consuming operations and the level of human intervention necessary to optimize MFs and fuzzy rules.

2.3. Weighted Support Vector Machines

Weighted Support Vector Machines (weighted SVMs) are also known as Fuzzy Support Vector Machines (FSVMs); a name proposed by Fan and Ramamohanarao (2005) as weight is effectively the fuzzy membership addressed for each training data point.[2] FSVMs were developed by Lin and Wang (2002) to enhance support vector machines (SVMs) abilities to reduce the effect of outliers and noise in data points. While SVMs theory has been demonstrated very powerful in solving classification problems (Burges, 1998), it has drawbacks. For example, SVMs treat all training points of a given class uniformly, however in many real world applications, not all training data point are equally important for classification purposes. To solve this problem, Lin and Wang (2002) applied a fuzzy member to each input data point in an SVMs, thus allowing different input data points to contribute differently to the learning decision surface. In such time series prediction problems, the older training points are associated with lower weights such that the effect of older training points can be reduced when the regression function is optimized.

Given a set S of labeled training data points associated with weights

$$(y_1, x_1, s_1), \ldots, (y_m, x_m, s_m) \tag{1}$$

where $x_i \in R^n$ is the input vector, $y_i \in R$ is the desired value and $\sigma \leqslant s_i \leqslant 1$ is a weight for $(x_i, y_i)(i = 1, \ldots, m)$ and a sufficiently small $\sigma > 0$ represents the lower bound of weighting data. The weighted SVMs for regression solves and optimizes:

$$\text{Minimize } \frac{1}{2} w \cdot w + C \sum_{i=1}^{l} s_i(\xi_i + \xi_i^*) \tag{2}$$

$$\text{Subject to } \begin{cases} y_i - (w \cdot \varphi(x_i) + b) \leqslant \varepsilon + \xi_i, \\ (w.\varphi(x_i) + b) - y_i \leqslant \varepsilon + \xi_i^*, \\ \xi_i, \xi_i^* \leqslant 0 \end{cases}$$

where C is a constant and $\varphi(x)$ is the high dimensional feature space, which is nonlinearly mapped from input space x. ξ_i and ξ_i^* represent upper and lower training errors, respectively. It should be noted that a smaller s_i reduces the effect of the parameter ξ_i in Eq. (2), such that the corresponding point $\varphi(x_i)$ is treated as less important.

To above optimization problem can be transformed into

$$\begin{aligned} \text{Maximize } W(\alpha) = & -\frac{1}{2} \sum_{i,j=1}^{l} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) K(x_i, x_j) \\ & - \varepsilon \sum_{i=1}^{l} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{l} y_i(\alpha_i - \alpha_i^*) \end{aligned} \tag{3}$$

$$\text{Subject to } \sum_{i=1}^{l} y_i \alpha_i = 0, \quad 0 \leqslant \alpha_i \leqslant s_i C, \ \ i = 1, \ldots, l$$

and the Karush-Kühn–Tucker condition is defined as

$$\overline{\alpha}_i(\varepsilon + \overline{\xi}_i - y_i + \overline{w} \cdot x_i + \overline{b}) = 0, \quad i = 1, \ldots, l, \tag{4}$$

$$\overline{\alpha}_i^*(\varepsilon + \overline{\xi}_i^* + y_i - \overline{w} \cdot x_i - \overline{b}) = 0, \quad i = 1, \ldots, l, \tag{5}$$

$$(s_i C - \overline{\alpha}_i)\overline{\xi}_i = 0, \quad i = 1, \ldots, l, \tag{6}$$

$$(s_i C - \overline{\alpha}_i^*)\overline{\xi}_i^* = 0, \quad i = 1, \ldots, l. \tag{7}$$

Point x_i with the corresponding $\overline{\alpha}_i^* > 0$ is a support vector. The other type of support vector, with corresponding $0 \leqslant \overline{\alpha}_i^{(*)} \leqslant s_i C$, lies on the ε-insensitive tube around the decision function. The one with corresponding $\overline{\alpha}_i^* = s_i C$ is outside the tube. An important difference between SVMs and weighted SVMs is that the points with the same value of $\overline{\alpha}_i^{(*)}$ may indicate a different type of support vector in weighted SVMs due to the factor $s_{\rm i}$ (Lin, 2004).

$K(x_i, x_j)$ in Eq. (3) is defined as the kernel function. The value of the kernel is equal to the inner product of two vectors X_i and X_j in the feature space $\varphi(x_i)$ and $\varphi(x_j)$, that is, $K(x_i, x_j) = \varphi(x_i) * \varphi(x_j)$. The chosen kernel function must fulfill Mercer’s condition, which determines whether a prospective kernel is actually an inner product in some space and guarantees that unique global optimal solutions are achieved (Burges, 1998). Several admissible kernel functions used today include the polynomial kernel, radial basis

[2] In this paper, to avoid confusion with the FL technique, the term “weighted SVM” is used.

function (RBF) and sigmoid kernel. However, the RBF kernel has been recommended for general users as a first choice due to its ability to analyze higher-dimension data, use of only one hyperparameter to search, and fewer numerical difficulties (Hsu et al., 2003).

In sequential learning and inference methods such as time series problems, where a point from the recent past may be given greater weight than a point from further in the past, function of time t_i can be selected as the weighted SVMs s_i scheme.

$$s_i = f(t_i) \tag{8}$$

with this scheme assuming the last point x_m as the most important and $s_m = f(t_m) = 1$ and the first point x_1 as the least important, and choosing $s_1 = f(t_1) = \sigma$ (Lin and Wang, 2002). Lin and Wang (2002) proposed two time functions, linear and quadratic, as shown in Eqs. (8) and (9). Both have been used by Khemchandani et al. (2009) on financial time series forecasting problems, who demonstrated their abilities to deliver better results than SVMs.

$$s_i = f_l(t_i) = at_i + b = \frac{1-\sigma}{t_m - t_1} t_i + \frac{t_m\sigma - t_1}{t_m - t_1} \tag{9}$$

$$s_i = f_q(t_i) = a(t_i - b)^2 + c = (1-\sigma)(\frac{t_i - t_1}{t_m - t_1})^2 + \sigma \tag{10}$$

Like SVMs, using weighted SVMs presents the user with a problem of how to set optimal parameters, as parameter selection affects weighted SVMs prediction accuracy. The three parameters that must be optimized when using RBF kernels include the penalty parameter (C), kernel parameter (γ) and lower bound of weighting data parameter (σ). To overcome this drawback, an optimization technique (e.g., fmGA) may be used to identify best parameters simultaneously (Cheng and Wu, 2009).

2.4. Fast Messy Genetic Algorithm

Fast messy genetic algorithms (fmGA) are a recently developed machine learning and optimization tool based on a genetic algorithm approach that can efficiently find optimal solutions for large-scale permutation problems. Such differ from simple genetic algorithms (sGAs), which describe possible solutions using fixed length strings. fmGA applies messy chromosomes to form strings of various lengths (Feng and Wu, 2006).

The fmGA was developed by Goldberg et al. (1993) as an improvement on the messy genetic algorithm. mGAs were initially developed to overcome the sGA linkage problem, which resulted from a parameter coding problem that could generate suboptimal solutions (Goldberg and Deb, 1991). However, mGA faced a problem as well. Goldberg et al. (1993) proposed three modifications in order to reduce the size of the initial population as well as mGA execution time initialization and primordial phase. Those modifications utilize probabilistically complete initialization (PCI) instead of partially enumerative initialization (PEI), use building block filtering (BBF), and take a more conservative approach to thresholding in tournament selection.

A messy chromosome is a collection of messy genes. A messy gene in fmGA is represented by the paired values, "allele locus" and "allele value". Allele locus indicates gene position and allele value represents the value of the gene in that position. Consider the two messy chromosomes as follows: chromosome C1: ((1 0) (2 1) (3 1) (1 1)) and C2: ((3 1) (1 0)) both represent valid strings with lengths of three. As the above example shows, messy chromosomes may have various lengths. Moreover, messy chromosomes may be either "over-specified" or "underspecified" in terms of encoding bit-wise strings. Chromosome C1 is an over-specified string, which has two different values in the gene 1 position. To handle this over-specified chromosome, the string may be scanned from left to right following the first-come-first-served rule. Thus, C1 represents bit string 011. On the other hand, a competitive template would be employed to evaluate an underspecified chromosome, such as C2. The competitive template is a problem-specific, fixed-bit string that is either randomly generated or found during the search process. As shown in Fig. 1, if the competitive template is 111, C2 represents bid string 011 by assigning corresponding allele values in the position of gene 2 from the competitive template to represent missing genes.

The fmGA contains two loop types – inner and outer. The process starts with the outer loop. Firstly, a competitive template is generated. The competitive template is a problem-specific, fixed-bit string that is randomly generated or found during the search process. Each outer loop cycle is called one "era", each of which iterates over the order k of processed building blocks (BBs). A building block is a set of genes, which are a subset of strings that are short, low-order and high-performance.

With the start of each new era, the three phase operations of the inner loop, including the initialization phase, the BBF or primordial phase, and the juxtapositional phase, are invoked. In the initialization phase, an adequately large population contains all possible BBs of order k. fmGA performs the PCI process at this stage, which randomly generates n chromosomes and calculates their fitness values. There are two operations in the primordial phase, namely BBF and threshold selection. In the primordial phase, 'bad' genes that do not belong to BBs are filtered out, so that, in the end, the resultant population encloses a high proportion of 'good' genes belonging to BBs. In the juxtaposition phase, operations are more similar to those of sGAs. The selection procedure for good genes (BBs) is used together with a cut-and-splice operator to

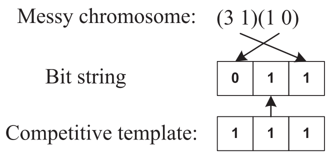

Fig. 1. Evaluation of an underspecified messy chromosome.

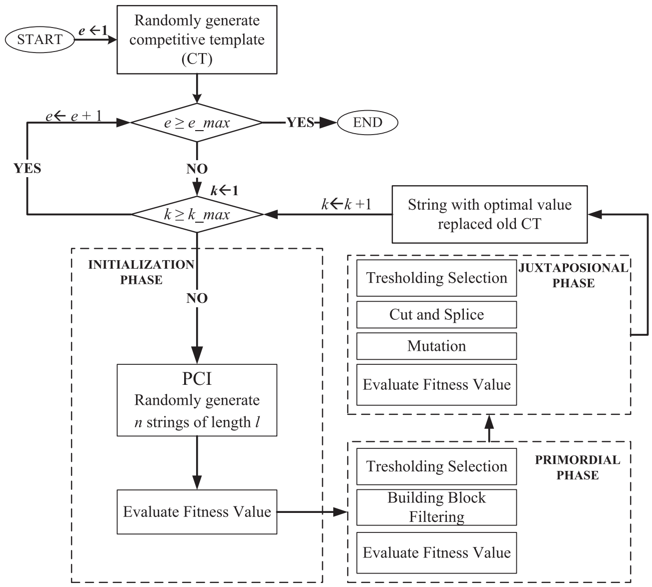

Fig. 2. fmGA organization.

form a high quality generation, which may contain the optimal solution.

Once the inner loop is finished, the next outer loop begins. The competitive template is replaced by the best solution found so far, which becomes the new competitive template for the next era. The whole process is repeated until the maximum number $k_{\max}$ is reached. The fmGA can also perform over “epochs”. This term is used to describe a complete process that starts from a first era and continues until $k_{\max}$. The best solution found in one complete process is passed to succeeding epochs though the competitive template. Epochs can be performed as many times as desired. The algorithm is terminated once a good-enough solution is obtained or no further improvement is made. Fig. 2 shows the organization of fmGA, where e represents epoch and k represents era.

3. Evolutionary Fuzzy Support Vector Machine Inference Model for Time Series Data

Once the disadvantages of a particular technique are understood and appreciated, they may be offset by the advantages of other techniques. The developed EFSIM$_T$ model is a hybrid AI system that fuses three different AI techniques; namely FL, weighted SVMs and fmGA. EFSIM$_T$, based on the FL paradigm, was developed to simulate the process of human inference. In this complementary system, FL deals with vagueness and approximate reasoning; weighted SVMs act as a supervised learning tool to handle fuzzy input–output mapping and focused on time series data characteristics; and fmGA works to optimize FL and weighted SVMs parameters.

The ability of FL to deal with vagueness and uncertainty depends heavily on the appropriate distribution of membership functions, number of rules and selection of proper fuzzy set operations. FL parameter construction is not easy, as they are problem oriented and rely heavily on expert knowledge. Therefore, weighted SVMs and fmGA have been introduced to resolve such issues.

EFSIM$_T$ architecture is shown in Fig. 3. In the figure, the fuzzy inference engine and fuzzy rules based on the conventional FL system are replaced by weighted SVMs. However, the generalizability and predictive accuracy of weighted SVMs are determined by searched problem parameters, including the optimal penalty parameter, kernel parameters and lower bound of the weighted data parameter. To overcome this shortcoming, EFSIM$_T$ utilizes fmGA to search simultaneously for optimum weighted SVMs parameters and FL parameters.

An explanation of major steps involved in EFSIM$_T$ is given below:

(1) *Training data*. The EFSIM$_T$ uses sequential data as training data. Sequential data reflects identified attributes, and training data are normalized into a (0, 1) range, which helps avoid attributes with greater numeric ranges dominating those with smaller numeric ranges, and also helps avoid numerical difficulties (Hsu et al., 2003). The function used to normalize data is shown in Eq. (10).

$$x_{sca} = \frac{(x_i - x_{\min})}{x_{\max} - x_{\min}} \tag{11}$$

(2) *Data weighting*. In this research, the LIBSVM developed Chang and Lin (2001) was embedded into the EFSIM$_T$ model. Each training data point was weighted to the time function using either a linear or quadratic function, as shown in Eqs. (8) and (9). The last point x_m, from the recent past, was treated as the most important and, as such, had a weighting value s_m set to 1. The first point x_1, from the most distant past, was treated as least important, and was given a weighting value s_1 of σ. In this step, the lower bound of weighting data parameters (σ) were generated randomly in range 0.1–1 and encoded by fmGA. Fig. 4 illustrates that points from recent past were given more weight than those from the more distant past in accordance with two different types of time function – linear and quadratic.

(3) *Fuzzification*. This is a process that converts each normalized input variable value from the first step into corresponding membership grades. MFs are used to represent the relationship and also to map normalized input variables to corresponding membership grades. This study used trapezoidal MFs and triangular MFs shapes (see Fig. 5) that, in general, may be developed by referencing summit points and widths (Ishigami et al., 1995). The summit and width representation method (SWRM) was used in this study to encode complete MF sets (see Fig. 5c) (Ko, 2002). Each normalized input pattern was converted to

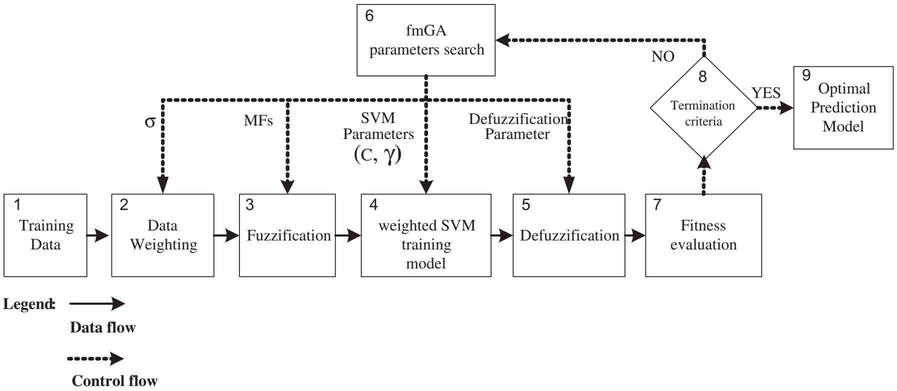

Fig. 3. EFSIM$_T$ structure.

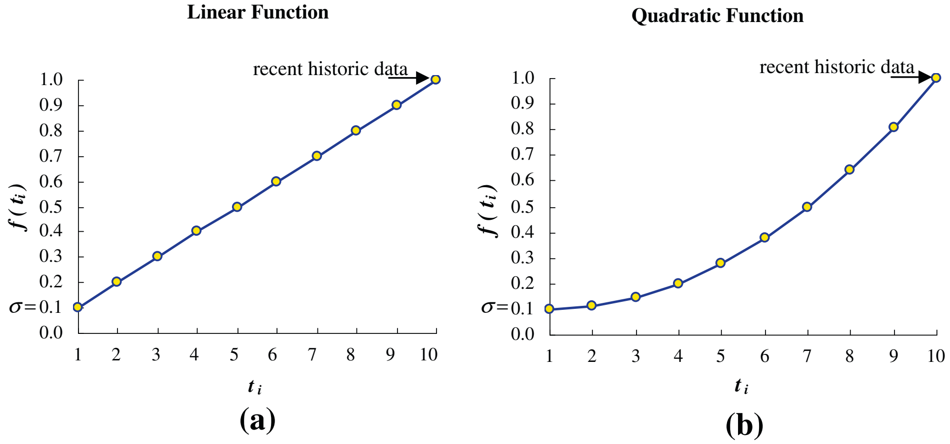

Fig. 4. Illustration curve of the time function: (a) linear and (b) quadratic (Lin and Wang, 2002).

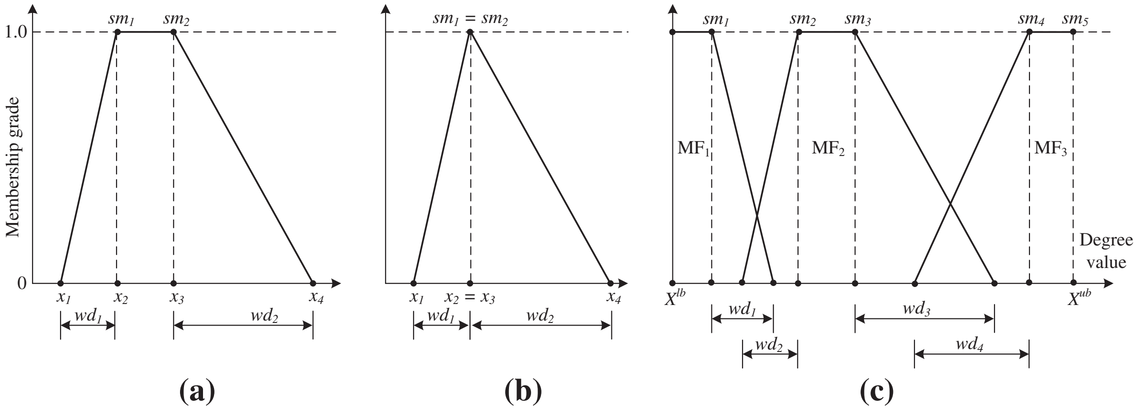

Fig. 5. Membership function: (a) trapezoidal; (b) triangular; (c) complete MF set (Ko, 2002).

membership grades corresponding to the specific MF set generated and encoded by fmGA. Fig. 6 illustrates the fuzzification process.

(4) *Weighted SVMs training model*. In this step, weighted SVMs developed based on SVMs are deployed to handle fuzzy input–output mapping. Fuzzification process output, in the form of membership grades, act as fuzzy input for weighted SVMs. Weighted SVMs train this dataset to obtain the prediction model. Weighted SVMs use penalty (C) and kernel parameters (γ), which are generated randomly and encoded by fmGA. This study used the RBF kernel as reasonable first choice (Hsu et al., 2003).

(5) *Defuzzification*. Once the weighted SVMs has finished the training process, output numbers are expressed in

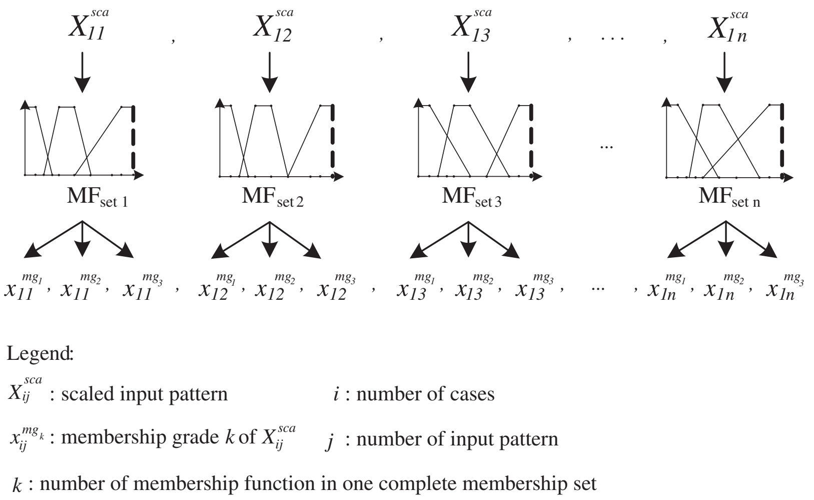
X_{11}^{sca} , X_{12}^{sca} , X_{13}^{sca} , . . . , X_{1n}^{sca}
MF_{set 1}
MF_{set 2}
MF_{set 3}
...
MF_{set n}
$x_{11}^{mg_1}$, $x_{11}^{mg_2}$, $x_{11}^{mg_3}$, $x_{12}^{mg_1}$, $x_{12}^{mg_2}$, $x_{12}^{mg_3}$, $x_{13}^{mg_1}$, $x_{13}^{mg_2}$, $x_{13}^{mg_3}$, ... , $x_{1n}^{mg_1}$, $x_{1n}^{mg_2}$, $x_{1n}^{mg_3}$
Legend:
X_{ij}^{sca} : scaled input pattern
i : number of cases
$x_{ij}^{mg_k}$: membership grade k of X_{ij}^{sca}
j : number of input pattern
k : number of membership function in one complete membership set

Fig. 6. Fuzzification process.

terms of the fuzzy set, and must be converted into a single real number. This conversion process is called defuzzification. Employing fmGA, the EFSIM$_T$ generates a random *dfp* substring and encodes it to convert weighted SVMs output. This evolutionary approach is simple and straightforward, as it uses *dfp* as a common denominator for weighted SVMs output.

(6) *fmGA parameter search.* The fmGA is utilized to search simultaneously for the fittest shapes of MFs, *dfp*, penalty parameter C, RBF kernel parameter γ and the lower boundary of weighting data parameter σ. The fmGA works based on the concept of genetic operations. For this reason, chromosome design plays a central role in achieving objectives. The chromosome that represents a possible solution for searched parameters consists of five parts: the MFs substring, *dfp* substring, penalty parameter substring, kernel parameter substring and lower bound of weighting data substring. Every substring has a specific length that should fit within certain requirements corresponding to the searched parameter, including length of decimal point string and upper and lower parameter bounds, among others.

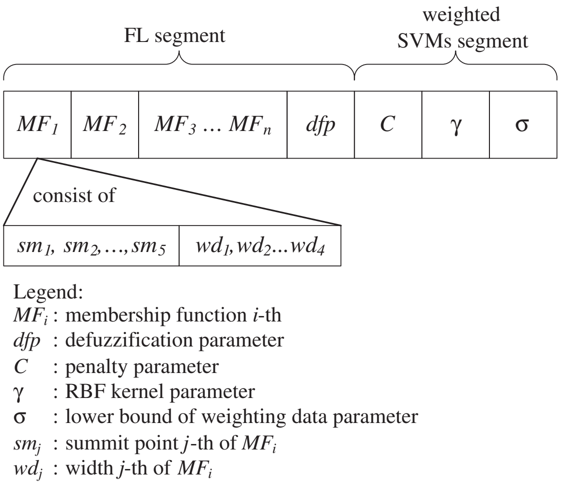
FL segment
weighted SVMs segment
MF_1
MF_2
MF_3 ... MF_n
dfp
C
γ
σ
consist of
$sm_1, sm_2, \ldots, sm_5$
$wd_1, wd_2 \ldots wd_4$
Legend:
MF_i : membership function i-th
dfp : defuzzification parameter
C : penalty parameter
γ : RBF kernel parameter
σ : lower bound of weighting data parameter
sm_j : summit point j-th of MF_i
wd_j : width j-th of MF_i

Fig. 7. EFSIM$_T$ chromosome structure.

The chromosome, as the model variable in EFSIM$_T$, is encoded into a binary string. Chromosomes consist of two segments, including FL and weighted SVMs. The FL segment contains MF and *dfp* substrings. The weighted SVMs segment contains penalty parameters C, kernel parameter γ from the RBF function and the lower boundary of weighting data parameter σ. Fig. 7 illustrates the chromosome structure.

As mentioned above, MF substrings are encoded using the SWRM method, which defines the distribution of uneven MFs by their summits and widths (see Fig. 5c). In Fig. 5a, summits of the trapezoidal MF are sm_1 and sm_2, whereas left and right widths are wd_1 and wd_2, respectively. A triangular MF may be regarded as a special trapezoidal MF case, in which $sm_1 = sm_2$. A complete MF set includes two shoulders. Fig. 5c shows the complete trapezoidal MF set, consisting of five summit points (sm_1, sm_2, sm_3, sm_4, sm_5) and four widths (wd_1, wd_2, wd_3, wd_4).

Applying the SWRM method, the required length of the MF binary substring RL^{MF} may be defined as follows:

$$RL^{\mathrm{MF}} = rn^{\mathrm{cMF}} \times (n^{sm} \times rl^{sm} + n^{wd} \times rl^{wd}) \tag{12}$$

where rn^{cMF} represents the required number of complete MF sets, n^{sm} represents the number of summits in a complete MF set, rl^{sm} represents the required length for a summit depending on the demand, n^{wd} represents the number of widths in one complete MF set, and rl^{wd} represents the required demand-dependent width. Considering that each input variable uses a common complete MF set to fuzzify crisp input data, rn^{cMF} is carried as follows:

$$rn^{\mathrm{cMF}} = \begin{cases} 1 & \text{if all input variables use a common complete MF set} \\ n^{iv} & \text{if each input variable uses its individual complete MF set} \end{cases} \quad (13)$$

where n^{iv} represents the number of input variables.

The *dfp* is a number searched by fmGA that will convert fuzzy output from the inference engine into crisp output. The required length rl^x of the *dfp* binary substring may be defined by adapting the variable mapping function Gen and Cheng (1997) from domain $[lb^x, ub^x]$, as follows:

$$2^{rl^x-1} < (ub^x - lb^x) \times 10^{rp} \leqslant 2^{rl^x} - 1 \quad (14)$$

where *rp* is the required places after the decimal point, and lb^x and ub^x are the lower and upper bound values of variable x. For the weighted SVMs parameter segment, which contains three substrings (i.e., the penalty parameter C substring, gamma γ substring and sigma σ substring), the required length of each binary C, γ and σ substrings, are also computed using Eq. (13). Table 1 summarizes parameter settings and number bits required for chromosome design.

(7) *Fitness evaluation*. Every chromosome that represents *MFs*, *dfp*, C, γ and σ is encoded and used to train the dataset. Model accuracy is obtained when a prediction model of the train dataset is gained. Each chromosome is further evaluated using a fitness function.

The fitness function was designed to measure model accuracy and the fitness of generalization properties (Ko, 2002). This function describes the fittest shape of MFs, optimized *dfp* number and weighted SVMs parameters. The fitness function integrates model accuracy and model complexity, as expressed in Eq. (14)

$$f^{fi} = \frac{1}{c^{aw} \times s^{er} + c^{cw} \times mc} \quad (15)$$

where c^{aw} represents the accuracy weighting coefficient, s^{er} represents the prediction error between actual output and desired output, c^{cw} represents the complexity weighting coefficient, and *mc* represents model complexity, which can be assessed simply by counting the number of support vectors.

Table 1
Summary of EFSIM$_\mathrm{T}$ parameter settings.

Parameter	Upper bound	Lower bound	Number of bits
MF set	–	–	27[a]
C	200	0	5
γ	1	0.0001	10
σ	1	0.1	10
dfp	1	0.5	9

[a] Number of bits required for one complete MF set.

(8) *Termination criteria*. The process is terminated when termination criteria are satisfied. While still unsatisfied, the model will proceed to the next generation. As the EFSIM$_\mathrm{T}$ uses fmGA, the termination criterion used here is either number of era (k) or number of epoch (e). The loop process continues when specified criteria are not met.

(9) *Optimal prediction model*. The loop stops once the termination criterion is fulfilled, which means that the prediction model has identified the input/output mapping relationship with optimal C, γ, σ and *dfp* parameters.

4. EFSIM$_\mathrm{T}$ for project cash flow prediction

This section validates the performance of the hybrid system EFSIM$_\mathrm{T}$ on a real world cash flow prediction problem. Data for the validation was obtained from a construction contractor in Taipei and cover high rise projects built by the contractor between 1996 and 2006. In this study cumulative cost-time curves was employed to model cash flow prediction.

As project cash flow problems are characterized by sequential data, the EFSIM$_\mathrm{T}$ holds the potential to solve such. In this simulation, the EFSIM$_\mathrm{T}$ adopted data reported by Liu (2006). The data pool contains percentage of expenditure cash flow (ECF) values taken from 13 similar projects. Liu (2006) split data into two groups, with 11 projects used as training data and two treated as testing data. Each project was divided into 20 sections, with each representing interval periods of 5% total project completion. To develop historical data, three sequential periods of ECF were used as input patterns, with the next used as output (see Table 2). Thus, for 11 projects, we had a total of 187 training data points, and 17 testing data points for each test case.

The accuracy of the proposed system was evaluated against other AI systems using RMSE and average error percentage. Table 3 shows the RSME and average error percentage per project comparison between the proposed EFSIM$_\mathrm{T}$ (linear and quadratic time functions) system and two other AI systems (SVMs and ESIM). In this study, as suggested by Hsu et al. (2003), the parameter setting for

Table 2
Example of sequential ECF training data from 1 project (Liu, 2006).

Case	Input pattern			Output
	1 1st period (%)	2 2nd period (%)	3 3rd period (%)	4th period (%)
1	0.34	4.97	6.38	7.21
2	4.97	6.38	7.21	9.71
3	6.38	7.21	9.71	15.99
...	...	...	...	...
16	78.16	82.04	90.65	95.23
17	82.04	90.65	95.23	100

Table 3
RSME and average error percentage per project comparison between EFSIM$_T$, SVM and ESIM.

	Test case							
	Project 27				Project 36			
	EFSIM$_T$		SVM	ESIM	EFSIM$_T$		SVM	ESIM
	Linear[a]	Quadratic[b]			Linear	Quadratic		
RMSE per project	0.0231	0.0231	0.0362	0.0360	0.0323	0.0314	0.0558	0.0481
Average error percentage per project (%)	1.54	1.53	3.02	2.66	2.40	2.24	4.41	3.98
C parameter	44	44	1	0	31	31	1	0
γ parameter	0.0030	0.0030	0.3333	0.7000	0.0109	0.0109	0.3333	1.0000
dfp parameter	0.9815	0.9821	–	–	0.9815	0.9815	–	–
σ parameter	0.6032	0.8534	–	–	0.7908	0.7908	–	–

[a] Linear time function.
[b] Quadratic time function.

SVMs, herein C and γ, were set to 1 and $\frac{1}{k}$, respectively, where k represents number of input patterns.

Average RMSEs achieved by EFSIM$_T$ linear, EFSIM$_T$ quadratic, SVMs and ESIM for both test cases were 0.0277, 0.0273, 0.0460, and 0.0421, respectively. The average error percentages for both projects achieved by EFSIM$_T$ linear, EFSIM$_T$ quadratic, SVMs and ESIM were 1.974%, 1.887%, 3.715% and 3.320%, respectively. In comparing the proposed system, the EFSIM$_T$ (using either linear or quadratic functions) was found to perform better than either SVMs or the ESIM. This is because the EFSIM$_T$ copes better with time series data characteristics inherent to cash flow data as well as with the complex relationships between input and output variables and the uncertainty condition during the construction phase. Moreover, it can be seen that the differences between results obtained by EFSIMT linear and quadratic function are not significant. This shows that there is still room for improvement to find a better time function to deal with project case flow prediction.

According to Kenley and Wilson (1986) and Kaka and Price (1991) to ensure proposed system reliability, an acceptable forecast range of error for the construction industry should be within 3% of the contract amount. Results obtained by EFSIM$_T$, both linear and quadratic, were within acceptable limits. In addition, these results are more desirable than those obtained using the other two methods, as average EFSIM$_T$ performance was significantly better than both SVMs and the ESIM. Therefore, construction project managers may select and apply the EFSIM$_T$ model to predict project cash flow during project implementation.

5. Conclusion

This research proposed EFSIM$_T$ as a hybrid AI system to facilitate a proactive approach to controlling project performance focused on cash flow prediction. The EFSIM$_T$ system was developed by fusing together FL, weighted SVMs and fmGA. FL was used to address vagueness and approximate reasoning; weighted SVMs were concerned with fuzzy input–output mapping and focused on time series data characteristics; and fmGA was deployed as an optimization tool to handle FL and weighted SVMs search parameters. The EFSIM$_T$ infused the advantages of several current AI methods to enhance overall model effectiveness and overcome weaknesses inherent in each individual model. Moreover, by fusing FL and weighted SVMs, the EFSIM$_T$ was able to overcome difficulties inherent in cash flow problems, such as the complex relationship between input and output variables and the uncertainty inherent to the construction phase.

The EFSIM$_T$ model searches all possible MFs, defuzzification parameters, and weighted SVMs parameters (i.e., C, γ, and σ). The developed model reduced significantly the level of human intervention required to determine MF shapes and distributions through questionnaire surveys and expert interviews. The developed model also successfully identified the optimum penalty parameter, kernel parameter and lower bound of weighting data for weighted SVMs. As such, the proposed system greatly decreases the effort required to find optimum FL and weighted SVMs parameters. This system may be used by professionals without domain or AI knowledge. Hence, the EFSIM$_T$ has great potential as a predictive tool for cash flow management to control project performance.

The level of accuracy of the EFSIM$_T$ was found to be within acceptable limits. Moreover, EFSIM$_T$ model performance was superior to both SVMs and ESIM. Simulation results demonstrate EFSIM$_T$ validity when used for cash flow prediction in construction projects. By applying EFSIM$_T$ to cash flow problems, project managers can gain advance knowledge of project cash flow trends. This model can greatly assist project managers to control cash flow, based on insight that gives project managers a better opportunity to develop strategies that reflect actual conditions in a quick and timely manner in order to ensure project progress and profitability.

References

Bojadziev, G., Bojadziev, M., 2007. Fuzzy Logic for Business, Finance, and Management, second ed. World Scientific, Singapore.

Boussabaine, A.H., Kaka, A.P., 1998. A neural networks approach for cost-flow forecasting. Construction Management and Economics 16 (4), 471–479.

Boussabaine, A.H., Elhag, T.M.S., 1999. Applying fuzzy techniques to cash flow analysis. Construction Management and Economics 17 (6), 745–755.

Burges, C., 1998. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2 (2), 121–167.

Cao, L.J., Chua, K.S., Guan, L.K., 2003. Ascending support vector machines for financial time series forecasting. In: Proceeding of 2003 International Conference on Computational Intelligence for Financial Engineering (CIFEr2003), Hongkong, pp. 317–323.

Chang, C.C., Lin., C.J., 2001. LIBSVM: A Library for Support Vector Machines. <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.

Cheng, M.Y., Wu, Y.W., 2009. Evolutionary support vector machine inference system for construction management. Automation in Construction 18 (5), 597–604.

Cryer, J.D., Chan, K.S., 2008. Time Series Analysis with Applications in R, second ed. Springer, New York.

Fan, H., Ramamohanarao, K., 2005. A weighting scheme based on emerging patterns for weighted support vector machines. In: Proceeding IEEE International Conference on Granular Computing, vol. 2 (2), pp. 435–440

Feng, C.W., Wu, H.T., 2006. Integrated fmGA and CYCLONE to optimize schedule of dispatching RMC trucks. Automation in Construction 15, 186–199.

Gen, M., Cheng, R., 1997. Genetic Algorithms and Engineering Design. John Wiley & Sons, New York.

Goldberg, D.E., Deb, K., 1991. mGA in C: A Messy Genetic Algorithm in C. IlliGAL Technical Report 91008. Department of General Engineering, University of Illinois at Urbana-Champaign, Urbana, Illinois.

Goldberg, D.E., Deb, K., Kargupta, H., Harik, G., 1993. Rapid, accurate optimization of difficult problems using fast messy genetic algorithms. In: Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 56–64.

Heshmaty, B., Kandel, A., 1985. Fuzzy linear regression and its applications to forecasting in uncertain environment. Fuzzy Sets and Systems 15, 159–191.

Hsu, C.W., Chang, C.C., Lin, C.J., 2003. A Practical Guide to Support Vector Classification, Technical Report. Department of Computer Science, National Taiwan University, Taipei, Taiwan.

Hwang, S., Liu, Y., 2005. Proactive project control using productivity data and time series analysis, computing in civil engineering 2005. In: Proceedings of the 2005 ASCE International Conference on Computing in Civil Engineering, Mexico, pp. 12–15.

Hwee, N.G., Tiong, R.L.K., 2002. Model on cash flow forecasting and risk analysis for contracting firms. International Journal of Project Management 20 (5), 351–363.

Ishigami, H., Fukuda, T., Shibata, T., Arai, F., 1995. Structure optimization of fuzzy neural network by genetic algorithm. Fuzzy Sets and Systems 71, 257–264.

Kaka, A.P., Price, A.D.F., 1991. Net cash flow models: are they reliable? Construction Management Economic 9, 291–308.

Kenley, R., Wilson, O.D., 1986. A construction project cash flow model – an idiographic approach. Construction Management Economic 4, 213–232.

Khosrowshahi, F., Kaka, A.P., 2007. A decision support model for construction cash flow management. Computer-Aided Civil and Infrastructure Engineering 22 (7), 527–539.

Khemchandani, R., Jayadeva, Chandra, S., 2009. Regularized least square fuzzy support vector regression for financial time series forecasting. Expert System with Applications 36 (1), 132–138.

Klir, G.J., Yuan, B., 1995. Fuzzy Sets and Fuzzy Logic: Theory and Applications. Prentice Hall PTR, Upper Saddle River, New Jersey.

Ko, C.H., 2002. Evolutionary Fuzzy Neural Inference Model (EFNIM) for Decision-making in Construction Management. Ph.D. Thesis, Department of Construction Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan.

Lin, C.F., Wang, S.D., 2002. Fuzzy support vector machines. IEEE Transactions on Neural Networks 13 (2), 464–471.

Lin, C.F., 2004. Fuzzy Support Vector Machines. Ph.D. Thesis, Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan.

Liu, C.L., 2006. Prediction Cash Flow for Construction Projects using Evolutionary Fuzzy Neural Inference Model. MS Thesis, Department of Construction Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan (in Chinese).

Martinez, C., Castillo, O., Montiel, O., 2008. Comparison between ant colony and genetic algorithms for fuzzy system optimization. In: Castillo, O. et al. (Eds.), Soft Computing for Hybrid Intel Systems. Springer-Verlag, Berlin Heidelberg, pp. 71–86.

Park, H.K., Han, S.H., Russell, J.S., 2005. Cash flow forecasting model for general contractors using moving weights of cost categories. Journal of Management in Engineering 21 (4), 164–172.

Refenes, A., Refenes, N., Bentz, Y., Bunn, D.W., Burgess, A.N., Zapranis, A.D., 1997. Financial time series modeling with discounted least squares back-propagation. Neurocomputing 14, 123–138.

Russell, J.S., 1991. Contractor failure: analysis. Journal of Performance of Constructed Facilities 5 (3), 163–180.

Russell, J.S., Jaselskis, E.J., Lawrence, S.P., 1997. Continuous assessment of project performance. Journal of Construction Engineering and Management 123, 64–71.

Sapankevych, N.I., Sankar, R., 2009. Time series prediction using support vector machine: a survey. In: Computational Intelligence Magazine, IEEE in Computational Intelligence Magazine, vol. 4 (2), pp. 24–38.

Touran, A., Atgun, M., Bhurisith, I., 2004. Analysis of the United States department of transportation prompt pay provisions. Journal of Construction Engineering and Management 130 (5), 719–725.

Zadeh, L.A., 1965. Fuzzy sets. Information and Control 8 (3), 338–353.

Zadeh, L.A., 1973. Outline of a new approach to the analysis of complex systems and decision processes. In: IEEE Transactions on Systems, Man, and Cybernetics, vol. 3 (1), pp. 28–44.