Automation in Construction 18 (2009) 597–604

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

Evolutionary support vector machine inference system for construction management

Min-Yuan Cheng [1], Yu-Wei Wu *

Department of Construction Engineering, National Taiwan University of Science and Technology, Taiwan

ARTICLE INFO

Article history:
Accepted 9 December 2008

Keywords:
Fast messy genetic algorithms
Support vector machine
Object-oriented system development

ABSTRACT

Problems in construction management are complex, full of uncertainty, and vary based on site environment. Two tools, the fast messy genetic algorithms (fmGA) and support vector machine (SVM), have been successfully applied to solve various problems in construction management. Considering the characteristics and merits of each, this paper combines the two to propose an Evolutionary Support Vector Machine Inference Model (ESIM). In the ESIM, the SVM is primarily employed to address learning and curve fitting, while fmGA addresses optimization. This model was developed to achieve the fittest C and γ parameters with minimal prediction error. This research further integrates the developed ESIM with an object-oriented (OO) computer technique to create an Evolutionary Support Vector Machine Inference System (ESIS). Simulations conducted to demonstrate the robustness of the model in application indicate that ESIS may be used as a multifarious intelligent decision support system in decision-making to help solve a wide range of construction management problems.

1. Introduction

Construction engineering comprises a number of disparate activities, which relate to and impact upon one another and are affected by various uncertainties, such as weather, geological characteristics, human judgment, and market fluctuation. Professional construction management is necessary for construction engineering to accomplish construction objectives efficiently and is essential to project success [1].

Construction management is a process of accomplishing construction objectives through the application of available resources. Due to uncertainties and the changing nature of the construction industry, practical construction management problems are complex and ill-structured [2]. Developing a deterministic mathematical model to solve problems of construction management is difficult and expensive. Approximate inference, which is fast and cost effective, represents a viable alternative.

Inference is the process of deriving new facts from previously known information. When known information changes, the inference process should adapt accordingly. Construction management problems are complex and full of uncertainties, vagueness and incomplete or inexact data. Thus, "known" information is subject to continuous change. Therefore, the inference process must be adapted to fit environmental realities [3]. Humans are able to learn and capable of processing complex problems, even in the face of uncertainty, imprecision, and incomplete information. Imitating the process of human inference offers an effective approach to solving construction management problems. Artificial intelligence (AI) relates to computer system designs that handle and attempt to resolve problems intelligently by emulating processes inherent in the human brain. As AI technology enhances the ability of computer programs to handle tasks for which humans are currently still better at handling [4], employing AI paradigms is appropriate in efforts to solve construction management problems [5].

Various scientific and engineering fields have been paying increasing attention in recent years to fusing different artificial intelligence (AI) paradigms. A number of studies have demonstrated that performances achieved by fusing different AI techniques are better than those achieved by employing a single conventional technique [6].

Support vector machines (SVM) and fast messy genetic algorithms (fmGA) represent recently developed AI paradigms. SVM were first suggested by Vapnik [7] and have recently been applied to a range of problems that include pattern recognition, bioinformatics, and text categorization. SVM classifies data with different class labels by determining a set of support vectors that are members of the set of training inputs that outline a hyper plane in a feature space. It provides a generic mechanism that fits the hyper plane surface to the training data using a kernel function. The user may select a kernel function (e.g. linear, polynomial, radial basis, or sigmoid) for the SVM during the training process, which identifies support vectors along the function surface. Using SVM presents users with the problem of how to set optimal kernel parameters. Therefore, obtaining SVM parameters must occur simultaneously. Proper parameter settings can improve SVM prediction accuracy, with parameters that should be optimized including penalty parameter C and kernel function parameters such as the γ of the radial basis function (RBF) kernel. In designing an SVM, one must choose a kernel function, set

* Corresponding author. #43, Sec. 4, Keelung Rd., Taipei, 106, Taiwan, R.O.C. Tel.: +886 2 27330004; fax: +886 2 27301074.
E-mail addresses: myc@mail.ntust.edu.tw (M.-Y. Cheng), D9305503@mail.ntust.edu.tw (Y.-W. Wu).
[1] #43, Sec. 4, Keelung Rd., Taipei, 106, Taiwan, R.O.C. Tel.: +886 2 27336596; fax: +886 2 27301074.

0926-5805/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.autcon.2008.12.002

kernel parameters and determine a soft margin constant C (penalty parameter). The Grid algorithm is an alternative to finding the best C and γ when using the RBF kernel function. However, this method is time consuming and does not perform well [8,9]. Fast messy genetic algorithms (fmGA) were developed by Goldberg et al. in 1993. Unlike the well-known simple genetic algorithm (sGA), which uses fixed length strings to represent possible solutions, fmGA applies messy chromosomes to form strings of various lengths. Its ability to identify efficiently optimal solutions for large-scale permutation problems [15] gives fmGA the potential to generate SVM parameters C and γ simultaneously.

The primary objective of this research work is to develop an object-oriented Evolutionary Support Vector Machine Inference System for solving construction management problems.

In order to achieve the primary goals, firstly, this paper fuses fmGA and SVM to develop an Evolutionary Support Vector Machine Inference Model (ESIM) that simultaneously searches for the fittest SVM parameters within an optimized legal model. Then, the ESIM is integrated with OO computer techniques using the corresponding system development process to develop an OO intelligent inference system. The proposed system's potential to apply in various construction management problems opens up a new agenda for future research.

2. Support vector machines approach

The theory that underlies support vector machines (SVM) represents a new statistical technique that has drawn much attention in recent years. This learning theory may be seen as an alternative training technique for polynomial, radial basis function and multilayer percept classifiers. SVM are based on the structural risk minimization (SRM) induction principle [10], which aims to restrict the generalization error (rather than the mean square error) to certain defined bounds. SVM have been shown to deliver higher performance than traditional learning machines and have been introduced as powerful tools to solve classification and regression problems.

For the classification case, SVM identify a separating hyper plane that maximizes the margin between two classes. Maximizing the margin is a quadratic programming (QP) problem that can be solved from its dual problem by introducing Lagrangian multipliers.

Linear programming (LP)-SVM [7,26] is an important innovation due to its linearity and flexibility when used on large datasets. The term "linear programming" means the algorithm is based on linear programming optimization. Many experiments have demonstrated that the efficiency and performance of LP-SVM exceeds that of QP-SVM for purposes that include solving problems with very large sample sizes [26], improving computational speeds [27], and reducing support vector numbers [28]. Recent research work has demonstrated the convergence of QP-SVM [29,30].

In most cases, identifying a suitable hyper plane in input space is an application that is overly restrictive in practical applications. The solution to this situation is to map the input space into a higher dimension feature space, and then identify the optimal hyper plane within this feature space. Without any knowledge of the mapping, an SVM locates the optimal hyper plane using dot product functions in feature space known as "kernels". The kernel trick based on the Mercer theorem is used in SVM to map input data into high-dimensional feature spaces, wherein simple functions defined on pairs of input patterns are used to compute dot products and design a linear decision surface [31]. For input space X, if there is a mapping $\phi: X \rightarrow H$ that maps any $x, z \in X$ into Hilbert space H then a kernel, $K: X \times X \rightarrow R$, is constructed as $K(x,z) = \langle \phi(x), \phi(z) \rangle_H$, where $\langle \cdot, \cdot \rangle_H$ is the scalar product operator in H. The kernel function K satisfies the Mercer condition: the kernel matrix formed by restricting K to any finite subset of X is positive semi-definite, and, hence, K is usually called the Mercer kernel. The Mercer condition is essential to kernel design, as it is the key requirement for a unique global optimal solution to the kernel-extended pattern analysis algorithms based on convex optimization (e.g., SVM) [32].

The solution of the optimal hyper plane can be written as a combination of a few input points known as "support vectors". Regression problem equations in SVM are the same as those of classification problems, with the exception of their respective target variables. By introducing the ε-insensitive loss function and modifying slightly the equation formation, the SVM theory can be readily applied into regression problems [11,12].

Using a regression problem example: Suppose we are given a training data set $\{(x_1, y_1), \ldots, (x_n, y_n)\} \subset N \times K$, where N denotes the space of input patterns $\boldsymbol{R}^k$. In ε-SVM regression [6,7], the goal is to find a function $f(x)$ that has, at most, ε deviation from the actual obtained targets y_i for all the training data. In other words, we do not care about error as long as it is less than ε and will reject any deviation greater than ε. An ε-insensitive loss function is shown as Eq. (1).

$$|\xi|_\varepsilon = \begin{cases} 0 & \text{if } |\xi| \le \varepsilon \\ |\xi| - \varepsilon & \text{otherwise} \end{cases} \tag{1}$$

Function (1) is used so that error is penalized only when it falls outside the ε-tube.

SVM regression may be made nonlinear by simply mapping training patterns (x_i) through a nonlinear transform ($\Phi: N \rightarrow F$) into some high dimensional feature space (F). As shown in Eq. (2), a best fitting function is estimated in feature space F, where "." denotes the dot product in feature space F.

$$f(x) = w \cdot \Phi(x) \pm b \tag{2}$$

To avoid over-fitting, one should add a capacity control term, which in SVM case results is represented by $\|w\|^2$. Formally, the SVM regression model can be written as a convex optimization problem by requiring Eq. (3):

$$\min_{w,b,\xi_i,\xi_i^*} \frac{1}{2} |\ w|^{\ 2} + C \sum_{i=1}^{N} \left(\xi_i + \xi_i^*\right)$$

$$\text{Subject to} \quad \begin{array}{l} y_i - \left(w^t \Phi(x_i) + b\right) \le \varepsilon - \xi_i \\ \left(w^t \Phi(x_i) + b\right) - y_i \le \varepsilon - \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \quad \forall i. \end{array} \tag{3}$$

The constant $C > 0$ determines the trade-off between the complexity of $f(\mathbf{x})$ and the amount up to which deviations larger than ε are tolerated. What makes SVM regression attractive is that a linear function can be estimated in the feature space, even though a nonlinear function was estimated in the original space.

Some kernel functions include polynomial, radial basis (RBF) and sigmoid kernel [13], which is shown as functions (4), (5), and (6). Kernel parameters in the kernel functions should be set properly in order to improve predictive accuracy.

Polynomial kernel:

$$k\left(x_i, x_j\right) = \left(1 + x_i \cdot x_j\right)^d \tag{4}$$

Radial basis function kernel:

$$k\left(x_i, x_j\right) = \exp\left(-\gamma |\ x_i - x_j|^{\ 2}\right) \tag{5}$$

Sigmoid kernel:

$$k\left(x_i, x_j\right) = \tan h\left(k x_i \cdot x_j - \delta\right) \tag{6}$$

This paper suggests that, in general, RBF is a reasonable first choice [33]. The RBF kernel maps samples nonlinearly into a higher dimensional space. Therefore, unlike the linear kernel, it can handle cases where

relationships between class labels and attributes are nonlinear. Furthermore, the linear kernel is a special case of RBF, as [34] shows that the linear kernel with a penalty parameter C has the same performance as the RBF kernel with some parameters $(C; \gamma)$. In addition, the sigmoid kernel behaves like RBF for certain parameters [35]. The second reason is that the number of hyper parameters that influences model selection complexity. The polynomial kernel has more hyper parameters than the RBF kernel. Finally, the RBF kernel presents fewer numerical difficulties. Moreover, we must note that the sigmoid kernel is not valid (i.e., not the inner product of two vectors) under certain parameters [7]. However, there are certain situations where the RBF kernel is not suitable. In particular, when the number of features is very large, one may just use the linear kernel.

3. Fast messy genetic algorithms approach

fmGA, developed by Goldberg et al. [14], can find efficiently optimal solutions for large-scale permutation problems [15]. The fmGA-based approach is known for its flexibility in allowing hybridization with other methodologies to obtain better solutions [16]. fmGA elements and processes are described briefly in the following:

3.1. Messy representation

In the fmGA, genes of a chromosome are 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. For example, two messy chromosomes ((3 1)(1 0)(2 1)(4 1)(5 0)) and ((2 1)(5 0)(3 1)(1 0)(4 1)) are both equivalent to the binary string 01110. In addition, chromosomes may have various lengths. For instance, chromosomes S1:((5 1)(1 1)(3 0)(1 0)(4 1)(3 1)(2 1) (4 0)(5 0)) and S2: ((3 0)(1 1)(5 0)) both represent valid strings of length five. As the above example shows, messy chromosomes may be "over-specified" and "underspecified" in terms of encoding bit-wise strings. Chromosome S1 is an over-specified string which has two different values in the positions of genes 1, 3, 4, and 5. To evaluate an over-specified chromosome like S1, the string may be scanned from left to right following the first-come-first-served rule. Thus, S1 represents the bit string 11011. On the other hand, a competitive template would be employed to evaluate an underspecified chromosome, such as S2. The competitive template is a problem-specified and fixed-bit string that is randomly generated or represents a solution found during the search process. As shown in Fig. 1, if the competitive template is 10111, S2 represents bid string 10010 by assigning corresponding allele values from the competitive template to represent missing genes.

3.2. Messy operators

Messy operators, which include cut–splice and mutation operators, are used as genetic operators in the fmGA. The cut–splice operator, similar to the crossover operator in the sGA, is used to recombine different strings to create new strings. The cut operator breaks a messy string into two parts using a cut probability $P_c = P_k(\gamma - 1)$, where P_k is a specified bit-wise cut probability and γ represents string length. String length correlates positively with probability that the string will be cut. Cut point is chosen randomly along the string. For example, if P_k equals 0.1, the P_c of the string ((2 0)(5 0)(3 1)(6 0)(5 1)) would be 0.3. A cut at point 3 would result in strings ((2 0)(5 0)(3 1)) and ((6 0)(5 1)). The splice operator joins two strings with a specified splice probability

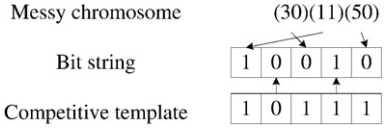

Fig. 1. Evaluation of an underspecified messy chromosome. [15]

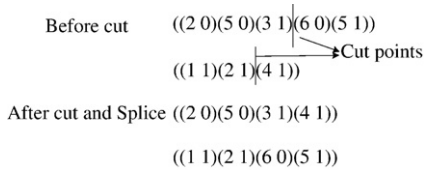

Fig. 2. Cut–splice operator. [15]

P_s. For example, as shown in Fig. 2, two strings ((2 0)(5 0)(3 1)(6 0)(5 1)) and ((1 1)(2 1)(4 1)) would be recombined to ((2 0)(5 0)(3 1)(4 1)) and ((1 1)(2 1)(6 0)(5 1)) after being cut and spliced. Goldberg et al. [14] proposed $P_k = 2/l$ and a maximum string length after being cut and spliced of $2l$, where l represents problem length. P_s is usually set to 1. The mutation operator perturbs messy chromosome allele values by switching them from 1 to 0 and vice versa with a predefined probability P_m. Fig. 3 demonstrates how the mutation operator works.

3.3. The fmGA organization

There are two loops, the outer loop and the inner loop, within the fmGA. The outer loop iterates over the order k of the processed Building Blocks (BBs). BBs are the schemata, which may be designated as "short", "low-order", or "high-performance". The order of BBs is defined as the number of fixed values in the chromosome. For example, if the BB is 011*1**, BB order equals 4. * symbolizes a "don't care" value. Every cycle of the outer loop is designated as one "era". When a new era starts, the inner loop, which includes the initialization phase, the primordial phase, and the juxtaposition phase, is invoked. The goal of the initialization phase is to create a population of strings containing all possible BBs of order k. fmGA performs the so-called "probabilistically complete initialization" process, which randomly generates n chromosomes of length γ, where $k < \gamma \leqq l$ and l represent problem length. The value of γ can be chosen arbitrarily, which is usually defined as $l - k$. The population size n can be approximated by using Eq. (7) for the binary-coded problem [17].

$$n = \frac{\binom{l}{k}}{\binom{l-k}{\lambda-k}} 2c(\alpha)\beta^2(m-1)2^k \tag{7}$$

Where, $c(\alpha)$ represents the square of a normal random deviate corresponding to tail-probability α, β represents the signal-to-noise ratio (i.e., the ratio of fitness deviation to the difference between two competing BBs). Variable m represents the number of BBs and k represents BB order. Those parameters may be set arbitrarily, in accordance with the problem. The primordial phase filters out the "bad" genes that do not belong to BBs, so that the resultant population encloses a high proportion of "good" genes belonging to BBs. Two operations, building-block filtering and threshold selection, are performed during the primordial phase. In the juxtaposition phase, those good genes (BBs) are combined by using selection and messy operators to form a high quality generation, which, perhaps, contains the optimal solution. Where the inner loop of the fmGA terminates, the outer loop of the fmGA begins, with processing BBs of order $k+1$. In addition, 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 required k_{max} is reached. In addition, fmGA can perform over "epochs",

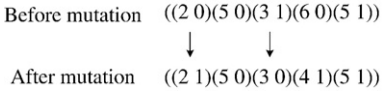

Fig. 3. Mutation operator. [15]

the term used to describe a procedure that starts from a first era and finishes with k_{max}. Epochs can be performed as many times as desired. In addition, the best solution found so far is that passed to epochs though the competitive template.

4. Evolutionary Support Vector Machine Inference Model

4.1. Chromosome design

To implement the proposed approach, this research used the RBF kernel function for the SVM because this function can analyze higher-dimensional data and requires that only two parameters, C and γ, be defined [18]. When the RBF kernel is selected, the parameters used as input attributes must be optimized using fmGA. Therefore, the chromosome comprises two parts, namely C and γ. The binary coding system is used to represent the chromosome.

4.2. Fitness function

The aim of the model is to obtain a solution that provides both a high degree of accuracy and the ability to be generalized to a broad problem set. While model accuracy in terms of input patterns can be improved by increasing the number of support vectors, an accurate model that is made to fit input patterns does not necessarily capture overall problem behavior well. In general, such models suffer from overfitting of input pattern data and a deterioration of generalization properties. Thus, the objective of ESIM is to use the fittest shapes of SVM with a minimum number of support vectors and optimal SVM parameters to preserve acceptable prediction accuracy in posed optimization problems. The objective function of model f^{ob} is to combine model accuracy and model complexity, as given in Eq. (8).

$$f^{ob} = c^{aw} \times s^{er} + c^{cw} \times mc \tag{8}$$

where c^{aw} is the accuracy weighting coefficient, s^{er} is the prediction error between actual output and desired output, c^{cw} is the complexity weighting coefficient, and mc is the model complexity, simply formulated by the number of SVM support vectors. The fitness function is the reciprocal of the objective function.

4.3. Model architecture

The following major steps (shown in Fig. 4) must be followed to establish an accurate fmGA-based parameter optimization model. A detailed explanation of such follows below:

(1) Training SVM. In this step, the SVM uses default parameters and a training dataset to train a prediction model.

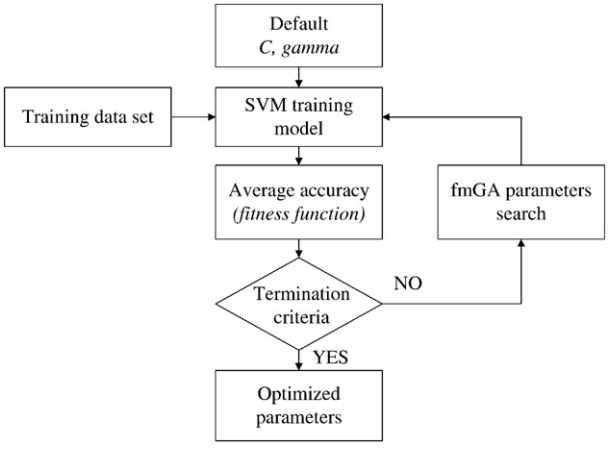

Fig. 4. Structure of the ESIM.

(2) Fitness evaluation. For each chromosome representing C and γ, a training dataset is used to train the SVM and calculate accuracy. When the accuracy is obtained, each chromosome is evaluated using a fitness function.
(3) Termination criteria. The process ends once termination criteria are satisfied. In the absence of such, the model proceeds to the next generation.
(4) fmGA parameters search. In this step, the model searches for better solutions by genetic operations.

4.4. Model adaptation process

The ESIM adaptation process is described by the following pseudo code algorithm.

```
Begin
Epoch = 1;
Generate the competitive template = random string;
While (not termination condition)//Outer Loop
{
    Repeat//Inner Loop
    {
        //Initialization Phase
        Era = 0;
        Probabilistic_Initialize(Pop(Era), Epoch);
        Evaluate(Pop(Era), template); //Evaluate fitness value
        //Primordial Phase
        While (not primordial termination condition)
        {
            Episode = 0;
            While (Episode<Episode_max(Era))
            {
                Thresholding_Selection(Pop(Era));
                Episode = Episode + 1;
            }
            Building-Blocks_Filtering(Pop(Era));
            Evaluate(Pop(Era),template); //Evaluate fitness value
        }
        //Juxtapositional Phase
        j = 0;
        While (not juxtapositional termination condition)
        {
            Thresholding_Selection(Pop(Era));
            Cut_and_Splice(Pop(Era));
            Mutation(Pop(Era));
            Evaluate(Pop(Era), template);
            j = j + 1;
        }
        template = Optimal_string(Pop(Era), Epoch);
        Era = Era + 1;
    }
    Epoch = Epoch + 1;
}
End
```

Probabilistic_Initialize: In this step, genes values are assigned with randomly generated with 0 or 1 to produce random variables, including C and γ to simulate a natural chromosome. Population size n can be approximated using Eq. (7) for the binary-coded problem.

Evaluate: The objective function of the model (f^{ob}) is a combination of model accuracy and model complexity, as given in Eq. (8). The fitness function is the reciprocal of the objective function.

Table 1
System functions [24].

Function (1)	Category (2)
Handle system parameters.	Evident
Handle solutions.	Evident
Handle different problems.	Evident
Execute the ESIM adaptation process.	Hidden
Setup model parameters.	Evident
Log completed information during model adaptation.	Hidden
Interrupt the model adaptation process.	Evident
Store optimum solution.	Hidden
Display information of the stored solutions.	Evident
Compute actual output of any case using any derived solution.	Hidden
Display actual output and desired output.	Evident
Plot the trend of actual output and desired output.	Evident

Thresholding_Selection: To restrict competition between building blocks with little in common, a generic thresholding mechanism was applied [14], where tournament selection between two strings is only permitted if they share a greater than expected number of genes in common. In random strings of two different lengths, λ_1, λ_2, the expected number of genes in common is $\theta = \left[\frac{\lambda_1\lambda_2}{l} + c'(\alpha')\sigma\right]$, where σ is the standard deviation in the number of genes that two randomly chosen strings of possibly differing lengths have in common and the parameter $c'(\alpha')$ is simply the ordinate of a one-sided normal distribution with tail probability α.

Building-Blocks_Filtering: A building block repetition factor γ (9) is used to acquire the desired building block.

$$\gamma = \left[\frac{\binom{\lambda_{i-1}}{\lambda_i}}{\binom{\lambda_{i-1}-k}{\lambda_i-k}}\right] \tag{9}$$

Cut_and_Splice: In this step, a cut probability is used to recombine different strings to create new strings.

Mutation: The mutation produces spontaneous random changes in various chromosomes, which protects against premature loss of important notations. For the ESVM, the purpose of mutation is to adjust the value of C and γ for better performance. It alters one or more genes with a probability (p^{ge}), which is smaller than or equal to the rate of mutation (p^{mu}). Mutation operation compares the gene's p^{ge} with p^{mu} bit by bit. If $p^{ge} \leq p^{mu}$, then the value of the gene will be altered.

5. System analysis and design

This research integrates the ESIM with OO computer technique to develop the Evolutionary Support Vector Machine Inference System. The system is developed based on three concepts, including the OO approach, incremental and iterative model, and Unified Modeling Language (UML). The OO approach is used to exploit the benefits of OO to develop the system. The incremental and iterative model is employed to increase developed system quality and reduce development time. UML is a standard OO specification language adopted by the Object Management Group (OMG) for specifying, visualizing, understanding, and documenting OO software. For that reason, the development procedure workflow is arranged to reflect the logical dependencies of UML artifacts and regular system development needs. Within a single development cycle, major tasks include analysis and design work.

5.1. System analysis

Analysis is the investigative process to determine essential system functions. The analysis phase of system development focuses on understanding and representing the requirements and concepts related to a system [19]. OO analysis focuses on identifying and describing required objects in the problem domain. The following activities are used in ESIS analysis:

5.2. Defining system requirements

System requirements describe the needs/desires of system users [20]. The primary goal of this activity is to identify and document such requirements. The aim of this research was to develop an intelligent inference system to solve problems related to construction management. Hence, the ESIS is designed to address the application needs of various users. System users are defined as those who want to solve construction management problems using a conceptual approach and may be project managers, engineers, schedulers, cost controllers, designers, contractors, architects, government officials, industry associations, or others who fit the description of a system user. ESIS integrates the ESIM through the use of OO computer techniques. System functions, designed to meet the abovementioned requirements, are summarized in Table 1. “Evident” indicates that users should be cognizant that the related function is performed, while “hidden” indicates that function execution happens outside users' field of awareness.

5.3. Defining system concepts

Software problems are inherently complex. Dividing the problem space into comprehensive units is a common strategy employed to deal with this complexity. In OO analysis, complexity dimensions are decomposed into concepts [19], which are defined as categories of ideas or things. This research work “divides and conquers” ESIM complexity by concepts. ESIS terminology and vocabulary are also clarified through this process.

In the present research, system concepts are represented using package diagrams, which are sets of related concepts or packages. It represents system concepts in a high level way within simple groups. ESIS is a complex system comprising multiple concepts. To reduce system complexity, this research assembled concepts together to partition the system into smaller subsets. Derived domain concepts for the system are shown in Fig. 5.

5.4. System design

A design describes how a system works and extends analyzed results to produce specifications for system implementation. During analysis, system development focuses on identifying software programs that need to be developed and implemented. In the system design phase, decisions are taken with regard to how requirements will be satisfied. OO system design focuses on defining logical software objects. The blueprints for implementing the ESIS are discussed in the following:

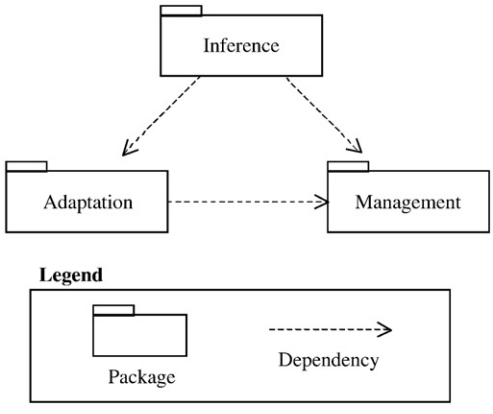

Fig. 5. System concepts [24].

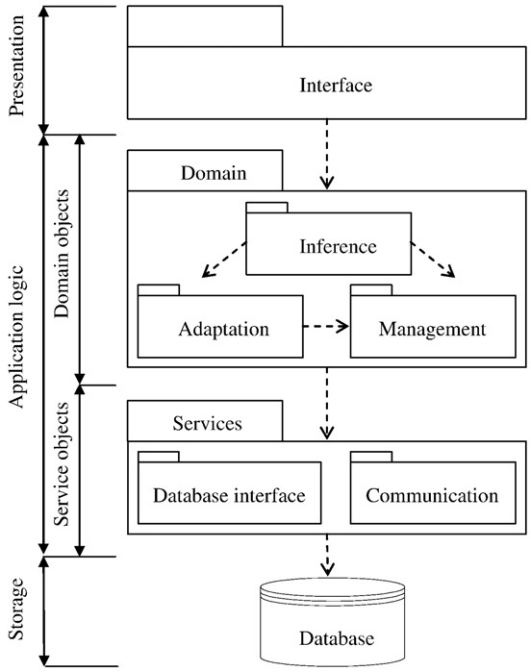

Fig. 6. System architecture [24].

5.5. Defining system architecture

Architecture, the organization or structure of a system, serves as the basis for system understanding, re-use, and evolution. It is also the abstraction of a system's implementation [21].

The common architecture of information systems, known as "three-tier" architecture, includes user interface, application logic, and storage. The logical architecture of the system is designed based on the three-tier architecture. The architecture package diagram for ESIS is shown in Fig. 6. In the diagram, packages illustrate groups of elements. This research work isolates the application layer in the middle layer. Therefore, these three tiers are independent so that the system enables to process different tasks simultaneously. This research divides the common three-tier architecture into multi-tiered architecture in order to decompose software complexity and exploit reusability advantages. In this paper, the application logic tier is decomposed into domain and service layers. The multi-tiered OO architecture can be characterized as having layers and partitions, with layers representing vertical tiers and partitions representing horizontal divisions of relatively parallel subsystems within a layer.

5.6. Defining software class

Class is a set of objects with semantics, properties, and behaviors common to other objects. In OO design, recognizing classes exposes commonalities to simpler designs within key abstractions and mechanisms [22]. How classes participate in the software solution explains details in the logical design of a system. A class diagram shows the existence of classes and their relationships. It is an abstract model that helps system participants to think about the system in the simpler terms of abstracted elements [23]. Fig. 7 is used to represent software classes and their relationships to the ESIS.

6. System validation

This section validates the performance of the ESIS. One artificial problem and a real construction management problem were examined to demonstrate the application potential of the developed system.

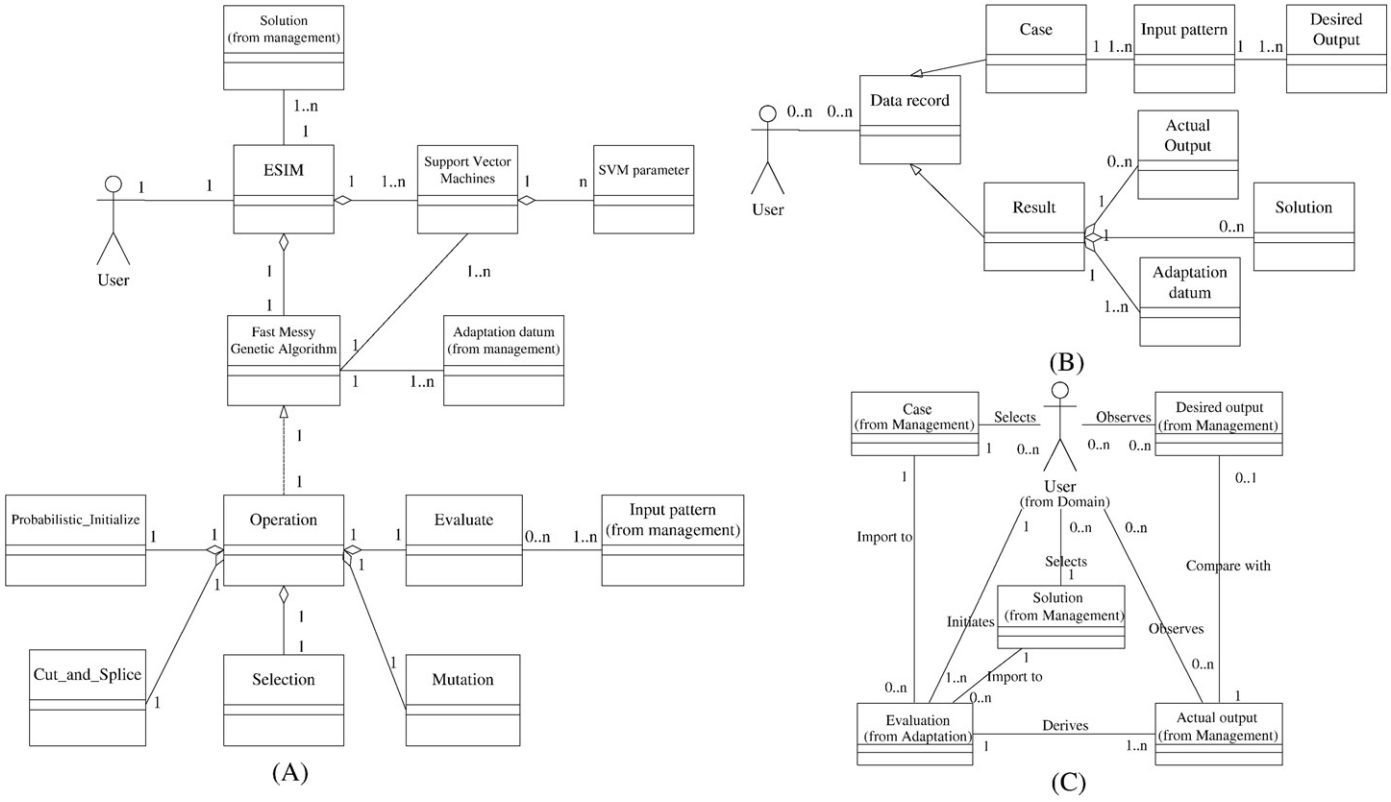

Fig. 7. (A) Class diagram: adaptation concept; (B) Class diagram: management concept; (C) Class diagram: inference concept.

Table 2
Simulation results of XOR.

Model	Classification	Regression
C	0	0
γ	0.8417	0.8233
Average accuracy	100%	92.4%
Computing time	87 s	47 s

6.1. Simulation 1: XOR problem

The exclusive-or (XOR), which is a simple nonlinear problem, has historically been considered as a good test of models. The XOR function maps two binary inputs to a single binary output as (Input 1, Input 2, Output) = (0,0,1), (1,0,1), (0,1,1), (1,1,−1). Table 2 illustrates the simulation results generated during the search process. In this table, the classification model quickly and successfully searches for the global optimum solution, with zero prediction error. In addition, the regression model searches for the optimal solution. According to results, the ESIS simultaneously obtained the global optimum solution with minimum prediction error.

6.2. Construction conceptual cost estimates

In the second simulation, the performance of ESIS in solving a real world problem is examined. For this simulation, 10 input patterns, 26 training data and 3 test data related to construction conceptual cost estimates are adopted from Hsieh [25]. The conceptual cost estimate is experience oriented. In conceptual planning phase, cost estimators can only estimate building cost according to preliminary design and project concepts. Under inadequate information circumstance, cost estimators refer to historical cases, and then judge conceptual cost based on their experiences. Nevertheless, building cost is effected by numerous factors. Some of these factors are full of uncertainty such as geological property and decorative class. Due to such complex and uncertain evaluation process, estimators evaluate building cost by experience cannot accurately evaluate the costs. As a result, present building cost estimates are rough. This validation aims to show that the ESIS can assist planners to estimate conceptual cost. According to the estimated cost, planners have more information to evaluate the project feasibility. Training data and test data were modified using a scale tool. Scaling data before applying SVM is very important. Sarle, W. S. [37] explains the reason for scaling data when using Neural Networks (NN). Most considerations also apply to SVM. The performances of different models in building cost estimation are listed in Table 3. Four prediction methods are compared in this table, including: 1) SVM $C=1$ and $\gamma=0.1$ (suggested by [33]); 2) Grid-SVM; 3) Evolutionary Fuzzy Neural Inference Model (EFNIM) [24]; 4)ESIM. The accuracy is evaluated using Root Mean Square Error (RMSE) and average deviation. Based on a literature review [25], approximately 25% is the acceptable deviation of present conceptual cost estimating.

Cheng [36] have demonstrated that EFNIM has better prediction performance than NN. The EFNIM employs GA to search simultaneously for the fittest MF shape, optimum FNN topology and optimum FNN parameters. However, an EFNIM shortcoming is its high computing time requirement. Grid-SVM is a traditional approach to search parameters. However this method requires a lot of computing time.

The result of ESIM is similar to EFINM and Grid-SVM as shown in Table 3. The average deviation of ESIM in this problem is 12.92% $((0.0112/0.7059+0.01/0.005+0.0949/0.2664)/3\times100\%=12.92\%)$. The RMSE of the ESIM is 0.0554. However, ESIM improves time requirement for developing the solution as well as prediction accuracy for the problem. Thus, planners can apply the ESIM in conceptual planning phase to estimate building cost. According to the inferred building cost, clients can make proper decisions to assess the project feasibility. In addition, the proposed method may assist clients to make various decisions, such as budgeting, tendering and awarding, and financial planning.

Table 3
Building cost simulation results.

Case no.	Building cost	SVM	Grid-SVM	EFNIM	ESIM
27	0.7059	0.6531	0.6944	0.6822	0.6947
28	0.5812	0.5310	0.5719	0.7194	0.5721
29	0.2664	0.3397	0.3611	0.2543	0.3613
C, γ		1, 0.1	5, 0.0251	–	5, 0.0257
RMSE		0.0596	0.0553	0.0813	0.0554
Average deviation		14.54%	12.92%	10.56%	12.92%
Computing time		~1 s	1893 sec	346 min	31 s

Real building cost is multiplied by 639 (in USD/m^2).

7. Conclusion

This research developed the ESIM by fusing a fast messy genetic algorithm together with an SVM. In the model, the SVM primarily addressed learning and curve fitting and fmGA primarily addressed optimization. The objective in developing this new model was to achieve concurrently optimal C and γ parameters and minimal prediction error.

In simulations run to test model validity, the developed intelligent inference model demonstrated learning, adaptation, fault tolerance, and self-organization abilities. It provides superior results to inference by overcoming existing bottlenecks, especially those requiring human intervention (e.g., mathematical equation development, expert interviews, and survey questionnaires). The ESIS, capable of running on a standalone PC, provides fast convergence speeds able to solve simulation problems and shows the potential for application among a wide range of prediction, synthetic evaluation, and control subject variables.

Since the ESIS adapts itself to input patterns, subjective human interventions are evaded. Also, the time and effort required for identifying the optimum system parameters are greatly improved by the system. The ESIS could be used as multifarious intelligent decision support system for decision-making to solve the manifold construction management problems.

This research showed experimental results using the RBF kernel. However, other kernel parameters may also be optimized using the same approach. The proposed approach may also be applied to support vector clusters. Because kernel parameters influence the predictive accuracy of support vector clusters employing different kernel functions, we can use the same parameter optimization procedures to improve support vector cluster accuracy.

References

[1] V.G. Bush, Construction Management: A Handbook for Contractors, Architects, and Students, Reston, Reston, Virginia, 1973, pp. 1–6.
[2] H. Li, Case-based reasoning for intelligent support of construction negotiation, Information & Management 30 (5) (1996) 231–238.
[3] I. Mareels, J.W. Polderman, Adaptive Systems: An Introduction, Birkhauser, Boston, Massachusetts, 1996, pp. 1–3.
[4] S. Haykin, Neural Networks: A Comprehensive Foundation, 2nd Ed. Prentice-Hall, New York, 1999.
[5] I.D. Tommelein, R.E. Levitt, B. Hayes-Roth, Site-layout modeling: how can artificial intelligence help, Journal of Construction Engineering and Management, ASCE 118 (3) (1992) 594–611.
[6] J.B. Yang, N.J. Yau, Integrating case-based reasoning and expert system techniques for solving experience-oriented problems, Journal of the Chinese Institute of Engineers 23 (1) (2000) 83–95.
[7] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.
[8] C.W. Hsu, C.J. Lin, A simple decomposition method for support vector machine, Machine Learning 46 (1–3) (2002) 291–314.
[9] C.L. Huang, C.J. Wang, A GA-based feature selection and parameters optimization for support vector machines, Expert Systems with Applications 31 (2) (2006) 231–240.

[10] Chun-Fu Lin, Fuzzy Support Vector Machines, Ph.D. dissertation, Dept. of Electrical Engineering, National Taiwan University, Taipei, Taiwan, 2004.
[11] H. Drucker, C. Burges, L. Kaufman, A. Smola, V.N. Vapnik, Support vector regression machines, Advances in Neural Information Processing Systems 9 (1996) 155–161.
[12] Pei Yi Hao, Jung Hsien Chiang, A fuzzy model of support vector regression machine, International Journal of Fuzzy Systems 9 (1) (March 2007) 45–50.
[13] C. Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery 2 (1998) 121–167.
[14] D.E. Goldberg, K. Deb, H. Kaegupta, G. Harik, Rapid, accurate optimization of difficult problems using fast messy genetic algorithms, Australian Electronics Engineering 27 (2) (1994) 56.
[15] Chung Wei Feng, Hsien Tang Wu, Integrating fmGA and CYCLONE to optimize the schedule of dispatching RMC trucks, Automation in Construction 15 (2) (2006) 186–199.
[16] T.M. Cheng, C.W. Feng, An effective simulation mechanism for construction operations, Automation in Construction 12 (3) (2003) 227–244.
[17] D.E. Goldberg, K. Deb, B. Krob, Don't worry, be messy, Proceedings of the Forth International Conference on Genetic Algorithms and their Applications, 1991, pp. 24–30, San Diego, USA.
[18] C.W. Hsu, C.J. Lin, A simple decomposition method for support vector machine, Machine Learning 46 (1–3) (2002) 219–314.
[19] C. Larman, Applying UML and Patterns: An Introduction to Object-oriented Analysis and Design, Prentice Hall PTR, Upper Saddle River, New Jersey, 1998.
[20] J.W. Satzinger, R.B. Jackson, S.D. Burd, System Analysis and Design in a Changing World, Course Technology, Cambridge, Massachusetts, 2000, pp. 67–75.
[21] D.F. D'Souza, A.C. Wills, Objects, Components, and Frameworks with UML: The Catalysis Approach, Addison-Wesley, Reading, Massachusetts, 1999.
[22] G. Booch, Object-oriented Analysis and Design with Applications, 2nd Ed. Benjamin, Redwood City, California, 1994.
[23] I. Jacobson, The Road to the Unified Software Development Process, Cambridge University Press, Cambridge, United Kingdom, 2000, pp. 103–108.
[24] Min Yuan Cheng, Chien Ho Ko, Object-oriented evolutionary fuzzy neural inference system for construction management, Journal of Construction Engineering and Management, ASCE 129 (4) (2003) 461–469.
[25] W.S. Hsieh, Construction conceptual cost estimates using evolutionary fuzzy neural inference model. MS thesis, National Taiwan University of Science and Technology, Taipei, Taiwan, 2002 (in Chinese).
[26] P.S. Bradley, O.L. Mangasarian, Massive data discrimination via linear support vector machines, Optimization Methods and Software 13 (2000) 1–10.
[27] J.P. Pedroso, N. andMurata, Support vector machines with different norms: motivation, formulations and results, Pattern Recognition Letters 22 (2001) 1263–1272.
[28] V. Kecman, I. Hadzic, Support vector selection by linear programming, Process of IJCNN 5 (2000) 193–198.
[29] I. Steinwart, Support vector machines are universally consistent, Journal of Complexity 18 (2002) 768–791.
[30] T. Zhang, Statistical behavior and consistency of classification methods based on convex risk minimization, Annals of Statistics 32 (2004) 56–85.
[31] A. Smola, B. Scholkopf, K.R. Muller, The connection between regularization operations and support vector kernels, Neural Network 11 (1998) 637–649.
[32] J. Shawe-Taylor, N. Cristianini, Kernel Methods For Pattern Analysis, 2004 Cambridge.
[33] Chih Wei Hsu, Chih Chung Chang, Chih Jen Lin, A Practical Guide to Support Vector Classification, Technical report, Department of Computer Science, National Taiwan University, July, 2003.
[34] S.S. Keerthi, C.J. Lin, Asymptotic behaviors of support vector machines with Gaussian kernel, Neural Computation 15 (7) (2003) 1667–1689.
[35] H.T. Lin, C.J. Lin, A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods, Technical report, Department of Computer Science, National Taiwan University, 2003.
[36] Min Yuan Cheng, Hsing Chih Tsai, Chien Ho Ko, Wen Te Chang, Evolutionary fuzzy neural inference system for decision making in geotechnical engineering, ASCE Journal of Computing in Civil Engineering 22 (4) (2008) 272–280.
[37] Sarle, W. S., Neural Network FAQ, Periodic posting to the Usenet newsgroup comp. ai.neural-nets, 1997.