Determining Attribute Weights in a CBR Model for Early Cost Prediction of Structural Systems

Sevgi Zeynep Doğan[1]; David Arditi, M.ASCE[2]; and H. Murat Günaydın[3]

Abstract: This paper compares the performance of three optimization techniques, namely feature counting, gradient descent, and genetic algorithms (GA) in generating attribute weights that were used in a spreadsheet-based case based reasoning (CBR) prediction model. The generation of the attribute weights by using the three optimization techniques and the development of the procedure used in the CBR model are described in this paper in detail. The model was tested by using data pertaining to the early design parameters and unit cost of the structural system of 29 residential building projects. The results indicated that GA-augmented CBR performed better than CBR used in association with the other two optimization techniques. The study is of benefit primarily to researchers as it compares the impact attribute weights generated by three different optimization techniques on the performance of a CBR prediction tool.

DOI: 10.1061/(ASCE)0733-9364(2006)132:10(1092)

CE Database subject headings: Construction costs; Cost estimates; Decision making; Decision support systems; Spreadsheets; Optimization models; Predictions.

Introduction

The construction industry utilizes experience. In construction management, knowledge of previous occasions is essential to providing solutions for current problems. Case based reasoning (CBR) has grown to be an artificial intelligence (AI) based method that offers an alternative for solving construction related problems that require extensive experience. Recent research demonstrated the potential benefits of this technique in construction management and its superior performance over other AI and traditional prediction techniques (Arditi and Tokdemir 1999a,b; Yau and Yang 1998). Further exploring CBR's capability in the construction management domain is a worthwhile task.

CBR has the ability to utilize existing data as cases. In other words, CBR can retrieve previously stored solutions from a case base to predict the outcome of a test case (Fig. 1). Constructing CBR systems requires a significant knowledge engineering effort (Cunningham and Bonzana 1999). The knowledge acquisition effort can be minimized by determining the most representative case attributes, optimizing the case base organization and case retrieval, and refining the process of similarity assessment (Jarmulak and Craw 1999; Jarmulak et al. 2000). Similarity assessment involves a systematic comparison of the attributes of a test case with the attributes of all cases in the case base. Its efficiency largely depends on what weights reflect the relative importance of attributes. The setting up of attribute weights can be done by a domain expert, where the expert manually chooses the relevant attributes and indicates their relative importance for similarity assessment, but this can be difficult and unstable because it is difficult to find the right expert who is knowledgable about these issues and because expert opinion is quite subjective. It is likely that different experts will disagree on the weights. It is easy, yet not optimal, to rely on the opinions of one single expert. One can however explore methods such as the analytical hierarchy process that is used to elicit this kind of information by pairwise comparisons and a series of mathematical manipulations that make sure no inconsistencies exist between the different pairwise comparisons. This is a well established method developed by Saaty (1980, 1994). One can also use the Delphi method whereby a panel of experts is used to reach consensus after several rounds of probing (Linstone and Turoff 1975; Adler and Ziglio 1996). Given the subjective nature of attribute weight generation by a human expert, applying an automated algorithm to establish attribute weights is attractive.

This paper attempts to assess the performance of a spreadsheet based CBR prediction model by testing the impact of attribute weights generated by three different optimization techniques, namely feature counting, gradient descent, and genetic algorithms (GA). The weight generation processes and the subsequent CBR calculations are presented in a simple spreadsheet format that is transparent and easy to use. The model is tested by predicting the cost of the structural system of residential building projects at an early stage. The results are compared and recommendations made.

CBR Spreadsheet Simulation

Many practitioners are familiar with spreadsheet applications. In this study, a spreadsheet simulation model of a CBR system was set up in Microsoft Excel. This spreadsheet model represents a

[1]Visiting Research Scholar, Dept. of Civil and Architectural Engineering, Illinois Institute of Technology, Chicago IL 60616. E-mail: dogan@iit.edu

[2]Professor, Dept. of Civil and Architectural Engineering, Illinois Institute of Technology, Chicago, IL 60616. E-mail: arditi@iit.edu

[3]Associate Professor, Dept. of Architecture, Izmir Institute of Technology, Izmir, Turkey.

Note. Discussion open until March 1, 2007. Separate discussions must be submitted for individual papers. To extend the closing date by one month, a written request must be filed with the ASCE Managing Editor. The manuscript for this paper was submitted for review and possible publication on January 27, 2005; approved on March 24, 2006. This paper is part of the ***Journal of Construction Engineering and Management***, Vol. 132, No. 10, October 1, 2006. ©ASCE, ISSN 0733-9364/2006/10-1092–1098/$25.00.

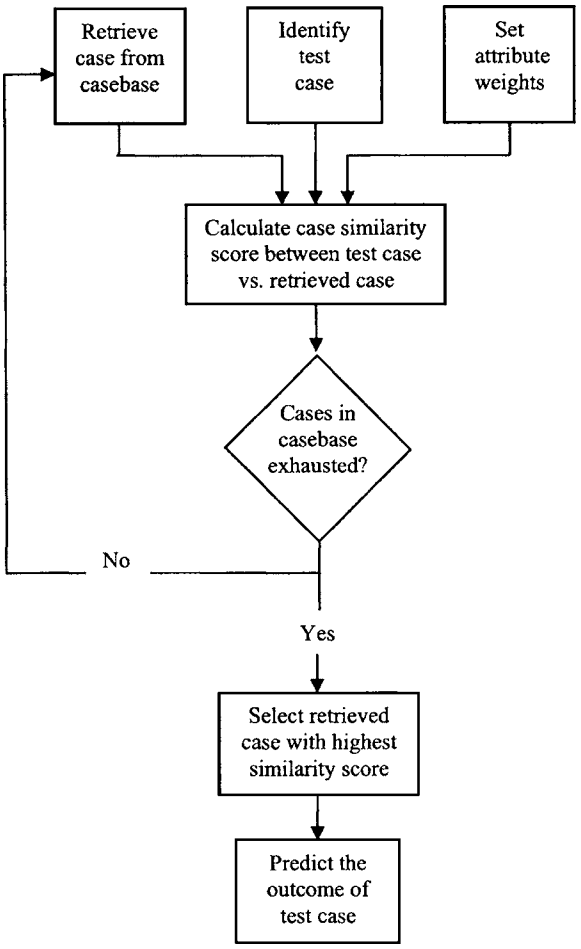

Fig. 1. Basic process of CBR

template for many prediction problems. The processing of the template involves six steps.

Step 1—*Organizing and formatting data:* The data are organized in the form of two matrices such as those presented in Fig. 2, one for the test cases and one for the input cases. Approximately 10% of all cases can be designated as test cases. The input and test cases are represented in rows and the input attributes are represented in columns. The output attribute is placed in a column next to the input attributes. The values of the attributes for each test and input case are represented, respectively, by I_{ik} and I'_{jk}, where I_{ik} represents the value of attribute k ($k=1,2,\ldots,p$) for test case i ($i=1,2,\ldots,m$), and I'_{jk} represents the same type of information for input cases j ($j=1,2,\ldots,n$). The weights of the attributes w_k ($k=1,2,\ldots,p$) are located at the top of the matrix in a row that corresponds to individual attributes. The way these weights are set is explained in Step 3. After formatting, semantic information is added to the data in the form of numerical and textual attribute values.

Step 2—*Calculating attribute similarities:* Attribute similarity functions are used to define how similar the attribute values are to each other. Attribute similarities are computed with respect to each test case versus every case retrieved from the input case base. Examples of textual and numerical similarity calculations are presented in Fig. 3. Attribute similarity is denoted by S_{ijk} where i=test case ($i=1,2,\ldots,m$), j=input case ($j=1,2,\ldots,n$), and k=attribute ($k=1,2,\ldots,p$).

Assuming that the value of the first attribute for the first test case I_{11} (in cell B5 in Fig. 2) is textual, its similarity with the

	A	B	C	D	E	F	G	H
1								
2	Weights	w_1	w_2	w_3	…		w_p	0
3	**Case No.**	TEST CASEBASE **Attributes**						**Output Attribute**
4		1	2	3	…		p	
5	Case 1	I_{11}	I_{12}	I_{13}	⋯		I_{1p}	O_1
6	Case 2	I_{21}	I_{22}	I_{23}	⋯		⋮	O_2
7	⋮	⋮						⋮
8	Case m	I_{m1}	I_{m2}	I_{m3}	…		I_{mp}	O_m
9								
10	**Case No.**	INPUT CASEBASE **Attributes**						**Output Attribute**
11		1	2	3	⋯		p	
12	Case 1	I'_{11}	I'_{12}	I'_{13}	⋯		I'_{1p}	O'_1
13	Case 2	I'_{21}	I'_{22}	I'_{23}	⋯		⋮	O'_2
14	⋮	⋮						⋮
15								
16	Case n	I'_{n1}	I'_{n2}	I'_{n3}	…		I'_{np}	O'_n
17								

Fig. 2. Formatting data to a case spreadsheet

corresponding attribute value I'_{11} (in cell B12 in Fig. 2) is established as follows:

If text in I_{11} appears to be exactly the same as text in I'_{11},

then similarity $S_{111}=1$, *or else* similarity $S_{111}=0$ (1)

(see Fig. 3 for spreadsheet calculations).

Assuming that the value of the third attribute for the first test case I_{13} (in cell D5 in Fig. 2) is numerical, its similarity with attribute value I'_{13} in the corresponding cell (D12 in Fig. 2) is established as follows:

	J	K	L	M	N	O	P	R	S
1									
2									
3	**Input Case No.**	**Attributes**							
4		1	2	3	…				p
5	Case 1	S_{111}	S_{112}	S_{113}	⋯				S_{11p}
6	Case 2	⋮	S_{122}		⋯				S_{12p}
7	Case 3		S_{132}						
8	⋮		⋮						
9									
10									
11									
12									
13	Case n	S_{1n1}	S_{1n2}		⋯				S_{1np}
14					…				

= MIN(D5,D$12)/MAX(D5,D$12)
Made once and copied to all cells with numerical information

=IF(B5=B$12,"1","0")
Made once and copied to all cells with textual information

Fig. 3. Attribute similarity matrix for Test Case 1 ($i=1$) (m similar matrices are generated, one for each test case)

J. Constr. Eng. Manage., 2006, 132(10): 1092-1098

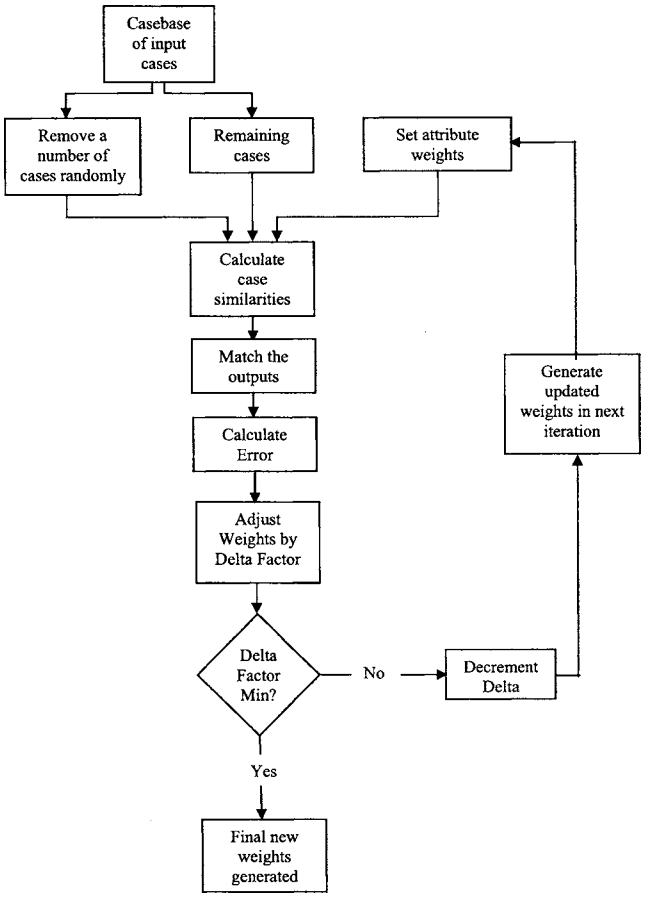

Fig. 4. Using gradient descent to optimize CBR weights

$$S_{113} = \frac{\min(|I_{13}|, |I'_{13}|)}{\max(|I_{13}|, |I'_{13}|)} \tag{2}$$

(see Fig. 3 for spreadsheet calculations).

Step 3—*Establishing attribute weights:* After all the attribute similarity values are calculated in $(n \times p)$ matrices, once for each test case (the matrix for Test Case 1 is presented in Fig. 3), the next step is to construct the weight vector that will be used in computing case similarities. Weights assign a value of importance to each attribute. In general, retrieval of the most relevant case is determined by the presence of a greater number of higher priority (more important) attributes matching between the test case and the retrieved case.

In this study, weights for attributes were computed by (1) the feature counting method, (2) the gradient descent method, and (3) GAs. In the feature counting method, the weight of each input attribute is entered as 1 into the CBR Excel model, implying that attributes have equal importance (Esteem 1996). In the absence of specific information, it is assumed that there is no reason for an attribute to be more important than another.

A CBR software called "Esteem" was used to implement the gradient descent weight generation method. The gradient descent weight generation method's basic algorithm is presented in Fig. 4 (Esteem 1996; Sutton and Barto 1998). Random cases are selected from the input case base, and the cases that are most similar to them (based on the initially set weights of the attributes) are found. Information on how much the weights of the attributes should be incremented or decremented is calculated considering these cases, based on how well the cases' outputs match. After examining several random cases, the resulting weights are adjusted by using a factor delta. The factor delta is then decreased, and the algorithm begins examining more random cases. This process continues until delta reaches a minimum value specified by the user. When the "arithmetic" method is chosen, delta is decremented by some value (which must be between 0 and 1) every iteration. When the "geometric" method is chosen, delta is multiplied by some factor (which must be between 0 and 1) every iteration. The user must also specify the starting and the final value of delta, the number of random cases that are examined every iteration, and an update parameter that specifies how quickly delta decreases from iteration to iteration. All parameters have default values which were used in this study: the geometric method was used with the starting and ending values of 0.5 and 0.02 for delta, respectively; the update parameter and the number of cases to be tested per each iteration were taken as 0.9 and 5, respectively. The weights generated by the gradient descent method were plugged into the CBR Excel model manually.

GA uses the method of evolution, specifically "survival of the fittest." The theory behind GA is that a population of certain species will adapt to live better in its environment after many generations of random evolution. Thus, GA first creates a population of possible solutions to the problem. Individuals in the population are then allowed to randomly breed, which is called crossover, until the fittest offspring (the one that solves the problem best) is generated. After a large number of generations, a population eventually emerges wherein the individuals will provide an optimum solution. For this study, a commercial GA software, Evolver, was used to find the optimum weights of the model. Evolver works as an add-in to Microsoft Excel (Palisade 1998). Weights generated by Evolver were plugged into the CBR Excel model manually.

Fig. 5 shows the flowchart of the GA optimization process used in this study. In order to use GA to generate weights, one of the cases in the input case base is removed and called an "evaluation case." The similarities between the attributes of the evaluation case and the corresponding attributes of the remaining cases are calculated by using Eqs. (1) and (2). Given the start-up assumption that attributes have equal importance, case similarities (CS) are derived between the evaluation case versus the remaining input cases by taking the average of all attribute similarities. The relationship that governs the similarity (CS) of the input case that has an output that is closest to the output of the evaluation case is plugged into the GA algorithm (Evolver) for maximization (for taking it closer to 1).

The Evolver optimization screen is shown in Fig. 6 with the adjustable cells containing the optimization variables (called attribute weights in the CBR system and chromosomes in GA terminology). In this study, the range of the attribute weights was set between 1 and 10, the default population size of 50 was used, and Evolver was run 15,000 times to find the optimum attribute weights that generated the maximum case similarity CS (closest to 1). This process was repeated as many times as the number of cases in the input case base by taking a different case out as the evaluation case at each cycle. The averages of the weights produced by GA at each cycle were used to run CBR in Step 4.

Step 4—*Calculating weighted case similarities*: Case similarities are computed for each test case with respect to each input case by using the attribute similarities calculated in Step 2 and the attribute weights generated in Step 3. For positive weights and normalized similarities, the weighted case similarities are always between 0 and 1, with a score of 1 indicating the case most similar to the test case and 0 the least. Weighted case similarities

J. Constr. Eng. Manage., 2006, 132(10): 1092-1098

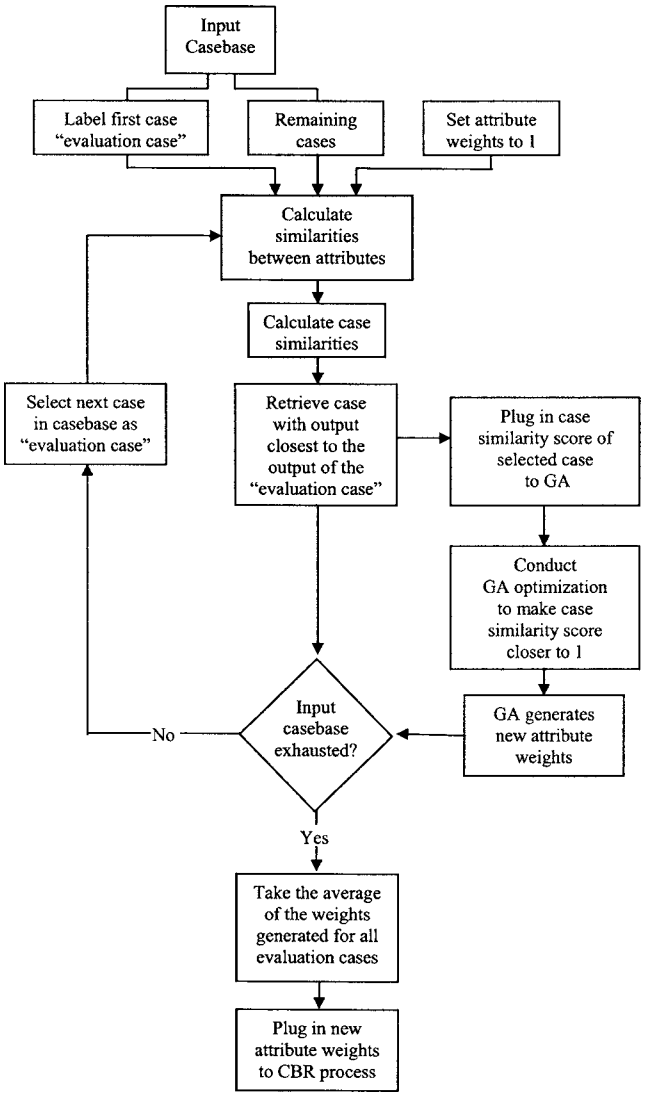

Fig. 5. Using GA to optimize CBR weights

are computed according to the following formula:

$$CS_{ij} = \frac{\sum_{k=1}^{p} S_{ijk} w_k}{\sum_{k=1}^{p} |w_k|} \quad (3)$$

for test case $i=(1,2,\ldots,m)$ and input case $j=(1,2,\ldots,n)$ for all attributes $k=(1,2,\ldots,p)$, where CS_{ij}=weighted case similarity between test case i and input case j over all the attributes k; S_{ijk}=similarity between test case i and input case j for attribute k; and w_k=weight of attribute k (see Fig. 7 for spreadsheet calculations).

Step 5—*Sorting weighted case similarities and corresponding outputs*: The highest weighted case similarity CS_{ij} for a test case i indicates the closest matching input case j in the case base. This operation is conducted (see Fig. 8) for each test case:

$$\max CS_{ij} = \max(CS_{i1}, CS_{i2}, \ldots, CS_{in}) \quad \text{for each } i(i=1,2,\ldots,m) \quad (4)$$

		T	U	V	Y	Z	AA
1							
2							
3		Input Case No.					Highest Score
4	Test Case No.	1	2	...		n	
5	Test Case 1	CS_{11}	CS_{12}	...		CS_{1n}	CS_{1x}
6	Test Case 2	CS_{21}	CS_{22}	...		CS_{2n}	⋮
7	Test Case 3	CS_{31}	⋮	...		CS_{3n}	= MAX (T5:Z5) Made once and copied down
8	⋮	⋮				⋮	
9		=(SUM (B$2*K5,C$2*L5,D$2*M5,E$2*N5,F$2*O5,G$2*P5))/(SUM(B$2,C$2,D$2,E$2,F$2,G$2) Made once and copied to all cells					
10							
11							
12	Test Case m	CS_{m1}	CS_{m2}	...		CS_{mn}	
13							

Fig. 6. Case similarity matrix for all test cases

Once the highest weighted case similarities are identified for respective test cases (see Column AA in Figs. 7 and 8), the corresponding case numbers and outputs are also listed (see Columns AB and AC in Fig. 8).

Step 6—*Calculating the error*: The random selection of the test set in Step 1 may affect the accuracy of testing. In other words, the test results are likely to change when different testing sets are selected. It is common practice to repeat the random project selection, training, and testing process several times and to pick the best results. The resulting outputs generated in the preceding step (Column AC in Fig. 8) are compared with the respective actual outputs (Column AD in Fig. 8, same as Column H in Fig. 2). The differences constitute the errors and are listed in Column AE in Fig. 8. The average of the error values of all test cases is the overall error of the CBR process.

Case Study

As an example application, the CBR Excel template was populated by data collected from residential building construction projects. The data were obtained from a research report that investigated the cost of the structural system in residential building construction projects undertaken in Turkey (Saner 1993). The unit

		AA	AB	AC	AD	AE
1						
2						
3		Highest Score	Case No.	Output Value	Actual Outputs for Test Cases	Error
4						
5	Test Case 1	CS_{1x}	x	O_x	O_1	E_{1x}
6	Test Case 2	CS_{2y}	y	O_y	O_2	E_{2y}
7	Test Case 3	⋮	⋮	⋮	⋮	⋮
8	⋮			=ABS((100-((AC5*100)/AD5))/100)		
9						
11	Test Case m	CS_{mz}	z	O_z	O_m	E_{mz}
12						$E_{average}$
				=AVERAGE(E_{1x} , E_{2y} , ..., E_{mz})		

Fig. 7. CBR outputs and calculating the error

J. Constr. Eng. Manage., 2006, 132(10): 1092-1098

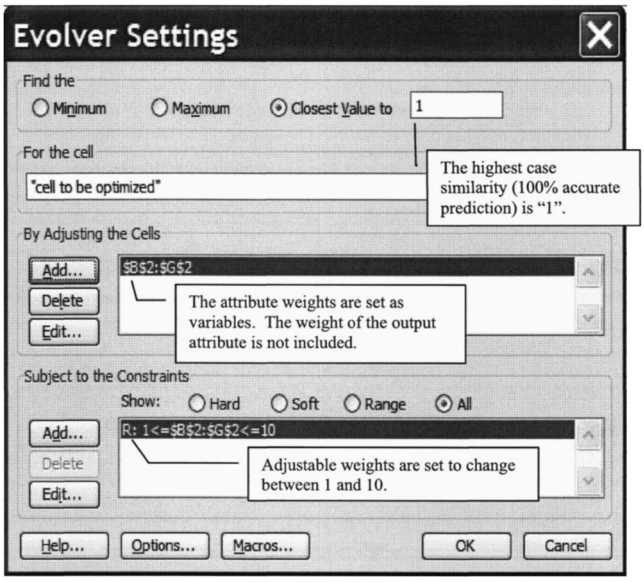

Fig. 8. Evolver optimization screen

cost of the structural system ($/m^2) was used as the *output* of the model. The structural system includes the superstructure and the foundations.

All possible *input* attributes that could affect the cost of the structural system and that could easily be identified in the early design stage were identified whereas those cost drivers that would not be readily available at the early design stage were left out (e.g., roof height, quality of the materials used, the ratio of the area of curtain walls to the total area of vertical construction, the ratio of the number of secondary beams to the total number of beams per floor of construction, etc). Only those attributes for which information was available in historical records were selected and those attributes whose values were identical for all cases were eliminated (e.g., year of construction: all projects were constructed in 1993; project delivery method: all projects were let by competitive bidding). The effectiveness of this type of prediction tool is enhanced by the selection of input attributes that are closely related to the output attribute. When this methodology was followed, the smallest number of input attributes turned out to be eight. The predominant cost drivers that were used as the *input attributes* (Table 1) included: (1) The total area of the building, which bears a strong linear relationship to the total cost of the building; (2) the ratio of the typical floor area to the total area of the building, which influences directly the cost of the components such as beams and columns of the load bearing frame; (3) the ratio of the footprint area to the total area of the building, which is correlated with the width and depth of the foundation system; (4) the number of floors, which has a direct effect on the structural design and consequently cost of columns; (5) the type of overhang, which may range between no overhang in the design to one-way or two-way designs, each alternative progressively increasing the cost of the structural system; (6) the location of the core of the building (i.e., vertical circulation system including stairs, elevators, and the service ducts), a central location requiring less cost than a side location which necessitates extra curtain walls to counteract torsion effects; (7) the type of floor, which includes cast in situ concrete floor systems or precast concrete structural units; and (8) the foundation system classified as pier, wall or slab foundations, each necessitating different amounts of concrete and reinforcement. The information relative to the textual attributes and the range of values relative to numerical attributes are presented in the last column of Table 1.

Table 1. Attributes Used in CBR Prediction Model

Input attribute number	Attribute	Range
1	Total area of the building	330–3,484 m^2
2	Ratio of the typical floor area to the total area of the building	0.07–0.26
3	Ratio of the footprint area to the total area of the building	0.07–0.30
4	Number of floors	4–8
5	Type of overhang design	No overhang, one-way or two-way overhang
6	Foundation system	Pier, wall, slab
7	Type of floor structure	Cast in situ concrete, precast concrete
8	Location of the core	At the sides, in the middle
Output	Cost of the structural system per square meter	$30–160/$m^2$

With the input attributes and the output defined, relevant data were then entered into the CBR Excel model using the procedure described in Fig. 2. A set of test cases were used to evaluate the effect of the attribute weights generated by all three methods, the data set of 29 projects being randomly split into an *input* set containing 24 projects, and a *test* set containing 5 projects. In other words, there were $i=1,2,\ldots,m$ $(m=5)$ projects that were used as test cases, $j=1,2,\ldots,n$ $(n=24)$ projects as input cases, $k=1,2,\ldots,p$ $(p=8)$ input attributes, and one output. The impact of the three sets of attribute weights was evaluated using the same test set of 5 projects.

The next step was to set up the Excel template to calculate attribute weights. The overall error obtained in CBR is a function of attribute weights. The attribute weights were generated by using (1) the feature counting method, (2) the gradient descent method, and (3) genetic algorithms. The attribute weights obtained (Table 2) were input into the CBR Excel application.

Results and Discussion

After the attribute weights were determined by using feature counting, gradient descent, and GA, the CBR Excel model was run and the performance of the model was evaluated. The CBR process of random selection of a 5 project test set, training and testing was repeated 10 times for different test sets and the best results were recorded. The results presented in Table 2 indicate that the GA-augmented CBR model yielded an average error of 16.23% whereas feature counting+CBR and gradient descent+CBR had average errors of 17.63 and 21.20%, respectively.

The setting up of the attribute weights in the feature counting method was straight forward in that all weights were taken as 1. In the gradient descent method, the experimentation between the arithmetic and geometric decrementation approach showed that the geometric approach resulted in better predictions. Default parameters were used for all other factors following the recommendations of the software developer (Esteem 1994).

GA optimization could have been performed with multiple evaluation cases. But the selection of 5 test cases out of a total of

J. Constr. Eng. Manage., 2006, 132(10): 1092-1098

Table 2. Optimized attribute weights and average error percentages for three methods

Weight generation method	Attribute weights								Average error in CBR prediction (%)
	Total area	Ratio of floor area to total area	Ratio of footprint area to total area	Number of floors	Overhang design	Core location	Floor type	Foundation system	
Feature counting	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	17.63
Gradient descent	0.0069	0.1885	0.1528	0.1427	0.1049	0.1560	0.0316	0.2161	21.20
Genetic algorithms	1.0000	2.0056	1.0010	9.9988	1.0031	1.0000	3.9999	1.0000	16.23

29 limited the number of cases in the input case base to as few as 24, which in turn necessitated the selection of very few evaluation cases (see Fig. 5), in this study only 1. Finally, in the GA optimization process, the weight of each attribute was constrained between 1 and 10. Using a range of 0–10 rather than 1–10 could have had the effect of eliminating certain attributes, hence making the process more efficient. It is worth exploring this issue in future research.

After the first cycle of the GA optimization process, the evaluation case was returned to the input case base and the next case picked for the next cycle of GA optimization. In other words, every input case in the input case base was used once as an evaluation case. As the similarity of two identical cases is indicated by 1 in the CBR system, the objective function of the GA optimization was set to make the case similarities closer to 1.

Three different approaches were experimented with in the GA optimization process to improve prediction accuracy. In the first approach, the objective of GA optimization at every cycle was to maximize the weighted case similarity of the input case that had the highest similarity with the evaluation case. In the second approach, the objective was to maximize the average weighted case similarity of all the 23 cases that were considered at each cycle. The third approach involved maximizing the weighted case similarity of the input case whose output was closest to the output of the evaluation case. The GA optimization process was performed for 24 cycles, each using these three approaches. The averages of the attribute weights determined in these 24 cycles were used in the CBR prediction model. The optimized attribute weights were calculated as follows:

$$w_k = \frac{\sum_{j=1}^{n} w_{jk}}{n} \tag{5}$$

where $k=1,2,\ldots,8$ attributes and $n=24$ input cases.

The performance of the optimized attribute weights were tested on the 5 test cases. Out of the three approaches, the third one performed best. The attribute weights presented in the last row of Table 2 were obtained by using this third approach.

One of the reasons why the average errors obtained (last column in Table 2) were not very low had to do with the nature of the output attribute. The output of the cases considered in this study was the unit cost of construction of the structural system and its value ranged between \$30 and \$160/m^2 (see Table 1). In order to have high prediction accuracy, one should have at least two or more cases with not only quite similar input attributes but also almost identical outputs, which is most improbable given the small number of cases (total 29) that were available for this study and the wide range of unit costs associated with the cases considered. The average errors reported in this study could have been lower had the output variables been binary or had there been a larger number of cases with an output attribute whose value varied in a smaller range.

Concluding Remarks

CBR was used in this study to develop a prediction model where attribute weights were generated by means of three optimization techniques, namely feature counting, gradient descent, and GA. The structure of the CBR model was simulated using an Excel spreadsheet to provide a transparent and simplified representation of this technique. The attribute weights that were generated by the three optimization techniques were then plugged into the CBR Excel model. The model was tested by using cost data pertaining to the early design parameters and unit cost of the structural system of 29 residential building projects. The results indicated that GA-augmented CBR performed slightly better than CBR used in association with the other two optimization techniques. Note that the differences in the average errors presented in Table 2 are not statistically significant.

The study demonstrated the practicality of using a spreadsheet in developing a CBR model for use in construction management. A spreadsheet simulation of an artificial neural network model developed by Hegazy and Ayed (1998) was the motivation of the study. A commercial CBR software using a spreadsheet based user interface helped to facilitate the simulation (Inductive 2000).

Even though the differences are small and not quite significant, it was not surprising to find out that feature counting+CBR did not generate predictions that are as strong as the prediction generated by the GA+CBR alternative because feature counting assigns equal weights to the attributes and therefore does not take into account the differences in importance of the attributes even though it is likely that such differences existed in the particular cases used in the study. But it was surprising to see that feature counting+CBR performed better than gradient descent+CBR. After all, gradient descent is a well established optimization technique that is routinely used in CBR systems (e.g., Esteem 1996). Although geometric descent was found to be more effective than arithmetic descent, the gradient descent experiments were conducted by using the default values of the parameters as recommended by Esteem (1996). Exploring the use of values other than the default values could possibly improve the performance of the gradient descent method, and in turn improve the predictions generated by gradient descent+CBR.

GA+CBR performed well despite the fact that the number of

J. Constr. Eng. Manage., 2006, 132(10): 1092-1098

cases in the case base was small and the output attribute was not binary. Both the GA optimization and the CBR prediction suffered from the fact that not many of the 29 cases considered in the study had input attributes and outputs that were close to each other. The likelihood of seeing stronger similarities is much higher if the number of cases is substantially higher than 29. Also, the study concentrated on optimizing attribute weights, while it should also be possible to improve attribute selection by using GA (Jarmulak and Craw 1999; Jarmulak et al. 2000), vary the number and combination of attributes, increase the number of cases considered, adjust the number and size of training and test sets, and try nondefault values for the parameters used in the various weight generation approaches. Considering the large number of alternative combinations of these variations, the study presented in this paper is limited to the basics and the experimentation with these variations are left for future research.

Despite the limitations cited earlier, the study is of benefit primarily to researchers as it illustrates the importance of attribute weights in the performance of a CBR prediction tool. It also indicates that it is worth experimenting with different weight optimization techniques rather than using the standard methodologies provided by a CBR tool.

References

Adler, M., and Ziglio, E. (1996). *Gazing into the oracle*, Jessica Kingsley Publishers, Bristol, Pa.

Arditi, D., and Tokdemir, O. B. (1999a). "Using case-based reasoning to predict the outcome of construction litigation." *Comput. Aided Civ. Infrastruct. Eng.*, 14(6), 385–393.

Arditi, D., and Tokdemir, O. B. (1999b). "Comparison of case-based reasoning and artificial neural networks." *J. Comput. Civ. Eng.*, 13(3), 162–169.

Cunningham, P., and Bonzano, A. (1999). "Knowledge engineering issues in developing a case-based reasoning application." *Knowledge-Based Systems*, 12(7), 371–379.

Esteem Software. (1996). *Esteem 1.4: Case based reasoning development tool*, San Mateo, Calif.

Hegazy, T., and Ayed, A. (1998). "Neural network model for parametric cost estimation of highway projects." *J. Constr. Eng. Manage.*, 124(3), 210–218.

Inductive Solutions, Inc. (2000). *Induce-It user manual*, New York

Jarmulak, J., and Craw, S. (1999). "Genetic algorithms for feature selection and weighting." *IJCAI-99 workshop on automating the construction of case-based reasoners*, S. S. Anand, A. Aamodt, and D. W. Aha, eds., 28–33.

Jarmulak, J., Craw, S., and Rowe, R. (2000). "Genetic algorithms to optimize CBR retrieval." *EWCBR 2000, LNAI 1898*, E. Blanzeri and L. Portinale, eds., Springer, Berlin, 136–147.

Linstone, H. A., and Turoff, M., eds. (1975). *The Delphi method: Techniques and applications*, Addison-Wesley, Reading, Mass.

Palisade Corp. (1998). *Evolver 4.0 for Excel reference manual*, Newfield, N.Y.

Saaty, L. T. (1980). *The analytic hierarchy process*, McGraw-Hill, New York.

Saaty, L. T. (1994). "Highlights and critical points in the theory and application of the Analytical Hierarchy Process." *Eur. J. Oper. Res.*, 74(3), 426–447.

Saner, C. (1993). "A proposal for cost estimation for structural systems 4-8 storey residential buildings." MSc thesis, Istanbul Technical Univ., Istanbul, Turkey.

Sutton, R. S., and Barto, A. G. (1998). *Reinforcement learning: An introduction*, MIT Press, Cambridge, Mass., ⟨http://www.cs.ualberta.ca/~sutton/book/8/node3.html⟩ (Feb. 1, 2004).

Yau, N. J., and Yang, J. B. (1998). "Case-based reasoning in construction management." *Comput. Aided Civ. Infrastruct. Eng.*, 13(2), 143–150.