Inductive Learning of Synthesis Knowledge

Yoram Reich
Steven J. Fenves
Engineering Design Research Center
Department of Civil Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
(yoram@cs.cmu.edu)
(412)-268-7864

International Journal of Expert Systems: Research and Applications, 5(4):275-297

Keywords: concept formation, synthesis, design, knowledge acquisition

Abstract: Even though synthesis is a crucial process in engineering design, synthesis knowledge is scarce. Previous studies in the use of machine learning techniques for design domains have concentrated mainly on learning analysis, rather than synthesis, knowledge. This study shows how synthesis knowledge can be acquired by concept formation. The paper evaluates the concept formation program COBWEB, discusses some of its shortcomings and presents extensions intended to broaden the applicability of COBWEB to design domains. The extensions are implemented in ECOBWEB—an enhanced version of COBWEB.

1 Introduction

Engineering design is the process of generating the description of an artifact that satisfies a *specification*. A design process can be roughly decomposed into distinct tasks of synthesis, analysis, redesign, and evaluation. *Synthesis* is the generation of candidate designs that are expected to satisfy the design specification. In real design problems, these candidates will partially violate some functional or behavioral aspect of the specification. These violation are detected by *analysis*, a functional evaluation or simulation of the behavior of the candidate design, to verify that it actually satisfies its intended function. *Redesign* is the task of modifying a candidate design to satisfy its specification and relevant design standards. *Evaluation* is the final design task which selects the design(s) from the set of candidates that satisfy the specification.

As an example, the specification for a new bridge may contain the location of the bridge, the number of lanes needed, and the length the crossing to be spanned. Synthesis generates several alternative candidates, such as cantilever, arch, or cable-stayed bridges, each with different dimensions and details. Analysis determines the degree to which these candidates satisfy the intended behavior specified by the relevant design standards, and redesign modifies the candidates accordingly. Finally, evaluation selects the best alternative for further, detailed design and, eventually, for construction.

The four tasks require distinct knowledge sources. The complexity and type of each knowledge source depending on the particular design domain. However, in most domains, high quality synthesis knowledge does not exist. This is in contrast to analysis knowledge, which is well developed within engineering disciplines because it is relatively easier to determine the behavior of artifacts than to understand how they are generated. In addition, historically, analysis has been perceived to be "scientific" whereas synthesis has been considered "artistic." Therefore, most engineering research has been directed towards providing better understanding of behavior revealed by analysis, rather than towards synthesis processes.

The above situation is unfortunate because synthesis is the most important process in design, as it deals with the creation of the new artifacts. The crucial question is, therefore, how synthesis knowledge can be *generated* to support the synthesis process.

Knowledge is not expected to be generated from scratch. For relatively mature disciplines, examples of existing designs are available. Since these designs are the products of exercising synthesis knowledge, they implicitly embed this knowledge. This, however, does not mean that synthesis knowledge is readily accessible.

The definitions of synthesis and analysis point out the inverse relation between the two processes. Therefore, a solution to the problem of generating synthesis knowledge is to use available examples of designs to generate an "inverse" to the analysis process (Whitehall et al., 1990). Machine learning techniques provide a means for generating the "inverse" knowledge.

The generation of the inverse mapping is not straightforward. Four characteristics of synthesis contribute to the difficulty of the learning task: (1) the evolving nature of engineering domains; (2) the extensive use of continuous values in engineering; (3) the nature of the mapping between specifications and design descriptions; and (4) the desirability to synthesize not one, but several candidates for a given specification. These characteristics translate into important concerns for selecting appropriate machine learning techniques for generating synthesis knowledge, elaborated below.

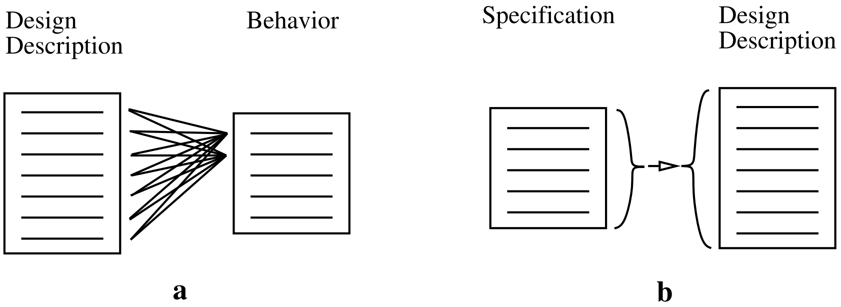


Figure 1: Mapping in analysis and synthesis

Mode of learning. Since design domains evolve, and since knowledge is continuously augmented from many sources, it is computationally inefficient to reprocess all previously accumulated information when new information needs to be learned. Clearly, when all the information is present, it is best to use it in a batch more. However, once we allow multiple methods to cooperate, or the user to intervene in the learning process, there is no way we can capture this without using an incremental system. Therefore, incremental learning techniques are preferred to batch techniques. Furthermore, in an evolving domain, the domain knowledge continuously changes, requiring a continuous adaptation of the learned knowledge. This adaptation is sometimes referred to in machine learning terminology as *tracking a concept drift* (Schlimmer and Granger, 1986). Adaptation to a changing domain is more important in design than the simple ability to learn incrementally. Very few of the incremental learning systems have demonstrated adaptation to date.

Type of properties. Real artifacts cannot be described by either nominal or continuous properties alone; rather, their description is always a combination of the two types (e.g., the type of bridge (nominal) and its span (continuous)). Therefore, common techniques such as symbolic learning (suitable for nominal properties) or statistics (suitable for continuous properties) are, by themselves, inappropriate. Rather, techniques that accommodate both property types are needed. The treatment of the two property types must be as similar as possible; otherwise, undesired learning bias may be introduced.

Type of mapping. One class of machine learning techniques uses the supervised concept learning method[1], where the knowledge to be learned is that of classification: a new object is to be classified into one of pre-defined classes. Therefore, the mapping between input and output is a *many-to-one* function. This is the mapping used in analysis, where the object is analyzed and its behavior is classified into a discrete range such as ‘acceptable’ or ‘unacceptable’ or a continuous range. Even if several analyses calculate different behaviors (e.g., structural and thermal), they can often be treated independently and can be modeled as several many-to-one functions. This mapping is summarized in Figure 1a. This definition of many-to-one mapping is different from the mathematical definition where several points in a domain set map onto one point in the range set. Although, in our situation they correspond; meaning, that a design can exhibit one behavior only.

Synthesis is different from analysis. In synthesis, the input is the specification, represented by

[1]In a supervised concept learning, a learner is given a set of pre-classified examples and it needs to create classification knowledge that will allow it to classify another example.

several properties, and the output is a full description of the artifact in terms of a much larger number of design description properties. The complete set of design description properties is needed to define a candidate design; therefore synthesis is a *few-to-many* function. This mapping is shown is Figure 1b. This definition of few-to-many is different than the mathematical definition where few points in a domain set can potentially map onto several points in the range set. Although, in our situation they correspond; meaning, that given one point in the domain set (i.e., a specification), we want to get several points in the range set (i.e., several candidates).

The distinction between the mappings used in analysis and synthesis is the crucial difference between the two processes. It influences the type of learning techniques that are suitable for the acquisition of synthesis knowledge. Whereas supervised concept learning techniques are appropriate for the acquisition of classification knowledge and therefore to many-to-one-mappings, unsupervised concept learning[2] is needed to support the acquisition of few-to-many mappings. This distinction is elaborated further in (Reich and Fenves, 1991). With the additional preference for incremental techniques presented previously, we conclude that incremental unsupervised concept learning is a preferred method for the acquisition of synthesis knowledge. These techniques are also called *concept formation* techniques (Fisher et al., 1991).

Performance task. The knowledge generated by existing machine learning techniques usually predict a single outcome. In the supervised concept learning method, the outcome is most often a class name and only rarely a continuous value. In the unsupervised method the outcome may be a list of properties characterizing a class. This outcome is inappropriate for synthesis. In synthesis, it is desirable, if not essential, that several alternative designs be generated for a given specification. Therefore, several outcomes must be possible from the use of the learned knowledge. This aspect makes further demands on the design of prediction methods for synthesis. However, from the perspective of testing these prediction techniques, it is easier to generate 'correct' results when predicting several outcomes, rather than one, since there is a high probability that the set of outcomes includes the 'correct' result. In the tests reported in this paper, a single outcome is generated to eliminate this testing problem. If we use other learning programs for comparison purposes with ECOBWEB, this elimination gives them an edge on ECOBWEB because the test is not performed on the task for which ECOBWEB was designed.

Figure 2 summarizes the contribution of the above four characteristics to the complexity of the task of learning synthesis knowledge. Item 1 in each of the issues denotes the common focus of machine learning, whereas item 2 denotes the focus of the techniques discussed in this paper.

This study makes several simplifying assumptions about the nature of design domains. The description of artifacts is limited to a list of property-value pairs. This list is called the *design description*. The list of properties in the design description is assumed to be fixed and pre-defined. The representation of the *specification* comprising the requirements, objectives and constraints for the artifact to be synthesized, is also limited to a fixed, pre-defined list of property-value pairs.

Using this representation, synthesis is a mapping from the specification properties to the design description properties. Therefore, learning synthesis knowledge is the *creation* of a mapping from the specification properties to the design description properties.

This study examines the use of a representative concept formation program COBWEB for the acquisition of synthesis knowledge and discusses some of its limitations for this task. Several extensions

[2] In an unsupervised concept learning, a learner is given a set of examples and it needs to create a classification of these examples. This classification can be used for many purposes, for example, prediction of missing properties.

mode of learning:	1. batch (most learning techniques) 2. incremental	increased complexity of learning
type of properties:	1. discrete values (early machine learning techniques) continuous values (statistics) 2. combination of two	increased complexity of learning
type of mapping:	1. many-to-one (supervised concept learning) 2. few-to-many (unsupervised concept learning)	increased complexity of learning
performance task:	1. prediction of a single outcome (supervised concept learning) 2. prediction of several alternate outcomes	increased complexity of learning

Figure 2: Characteristics influencing synthesis learning

embedded in an enhanced system, called ECOBWEB (Enhanced COBWEB), are described and tested against the behavior of the original system. ECOBWEB has been used as an important building block in a larger system that learns to design cable-stayed bridges (Reich, 1991).

In contrast to the work presented in this paper, most of the studies on the use of learning in design deal with analysis rather than synthesis knowledge (Arciszewski et al., 1987; Lu and Chen, 1987; Mackenzie and Gero, 1987). Other studies dealing with learning synthesis knowledge assume the existence of some knowledge and refine it (Mitchell et al., 1985), compile it into a more efficient form (Araya and Mittal, 1987), or build derivations from it (Mostow, 1989).

The remainder of this paper is organized as follows. Section 2 describes the domain of Pittsburgh bridges used as one of the tests of COBWEB and ECOBWEB. Section 3 reviews COBWEB and discusses its operation in design domains. Section 4.1 outlines the testing procedures used to evaluate COBWEB's learning performance. Section 4.2 reports test results on the use of COBWEB for the acquisition of synthesis knowledge. Section 5 outlines the extensions implemented in ECOBWEB and reports the results of these extensions on the same domain. Section 6 concludes the paper.

2 The domain of Pittsburgh bridges

The Pittsburgh bridge domain is used as a representative design domain to test COBWEB and ECOBWEB. This section describes the domain. The domain contains simplified descriptions of 108 bridges built in Pittsburgh since 1818[3]. Each example is described by 12 properties, 7 for the specification and 5 for the design description[4]. The task associated with this domain is the following: generate knowledge by

[3]The complete data with the documentation was donated to the *repository of machine learning databases* at the University of California, Irvine. This database and others can be accessed by ftp to ics.uci.edu, using anonymous as username and password. The files are in the /pub/machine-learning-databases directory.

[4]The 7-to-5 mapping in the domain of Pittsburgh bridges is not the typical few-to-many mapping in synthesis. This results from the simplification of the design description part of the examples, caused by missing data that would fully describe the designs. In another domain used to test ECOBWEB, the number of specification properties was 12 and the number of design description properties was 25. In the tests performed in that domain, four important specification properties were sufficient to

learning from a subset of the examples and use it to predict the 5 design description properties from the 7 specification properties of other examples.

The specification properties describe the location of the bridges and the objectives, requirements and constraints:

(1) The IDENTIFIER and the NAME properties identify the examples and are not used in the classification.
(2) The RIVER and LOCATION properties specify the location of the bridge.
(3) The PERIOD property specifies the time when the bridge was built.
(4) The PURPOSE property specifies whether it is a highway or railway bridge.
(5) The LANES property describes the number of lanes for a highway bridge or tracks for a railway bridge.
(6) The LENGTH property is the total length of the crossing.
(7) The CLEAR-G property specifies whether a vertical navigation clearance requirement was enforced in the design or not.

The design description properties specify the configuration of the actual bridges:

(1) The T-OR-D property specifies the vertical location of the roadway on the bridge: within the structure (through) or on top of it (deck).
(2) The MATERIAL property specifies one of the following: wood, iron, or steel.
(3) The SPAN property is the length of the main span of the bridge.
(4) The REL-L property is the relative length of the main span to the total crossing length.
(5) The TYPE property describes the type of the bridge, which may be one of the following: simple, continuous or cantilever truss; arch; suspension; or wood.

Table 1: Examples of bridge descriptions

	property	Example 1	Example 2	Example 3
specification	NAME[1]	16th St. B.	Fort Duquesne B.	Penn. Turnpike B.
	IDENTIFIER[1]	E19	E84	E88
	RIVER	Allegheny	Allegheny	Allegheny
	LOCATION	29	24	37
	PERIOD	Craft	Modern	Modern
	PURPOSE	Highway	Highway	Highway
	LANES	2	6	4
	LENGTH	Medium	Short	Long
	CLEAR-G	Not-governing	Governing	Not-governing
design description	T-OR-D	Through	Through	Deck
	MATERIAL	Wood	Steel	Steel
	SPAN	Medium	Medium	Long
	REL-L	Small	Full	Full
	TYPE	Wood	Arch	Continuous-Truss

[1] These properties are not used in the learning process.

Three examples of bridges are illustrated in Table 1. Table 2 provides a summary of the property-values describing the example set. The top line in each sub-table provides the property names and the frequency with which they appear in the example descriptions. The following lines provide the

obtain good synthesis performance (Reich, 1991).

Table 2: Summary of property-values

Specification

RIVER	108	**LOCATION**	107	**PERIOD**	108	**PURPOSE**	108
Youghiogheny	3	Varies from 1 to 52		Craft	18	Highway	71
Ohio	15			Emerging	15	RR	32
Monongahela	41			Mature	54	Aqueduct	4
Allegheny	49			Modern	21	Pedestrian	1

LANES	92	**LENGTH**	81	**CLEAR-G**	106
1	4	Short	12	G	80
2	61	Medium	48	N	26
4	23	Long	21		
6	4				

Design Description

T-OR-D	102	**MATERIAL**	106	**SPAN**	92	**REL-L**	103	**TYPE**	105
Through	87	Steel	79	Long	30	Full	58	Continuous-Truss	10
Deck	15	Iron	11	Medium	53	Medium	15	Cantilever-Truss	11
		Wood	16	Short	9	Small	30	Arch	13
								Simple-Truss	44
								Suspension	11
								Wood	16

property-values and their frequencies.

The property values provided in Tables 1 and 2 are all nominal. The discretization of the continuous properties (*PERIOD, LENGTH, SPAN, and REL-L*) was done in two stages. First, the domain of each continuous property was subdivided into pre-defined ranges. Second, the actual continuous values were assigned the name of the range that contained them. This discretization was performed for purposes of clarity and to allow programs that cannot handle continuous properties to use the data.

A property that plays a major role in the classification of bridges is the *PERIOD* property. In the different periods, different technologies and sources of knowledge were used and different requirements enforced, giving rise to increased correlations between the properties of bridges from the same period. For example, after the mid-1800's it became mandatory to design bridges so as to provide sufficient vertical clearance for navigation. In a flat terrain, this requirement correlates with choosing a *through* structure for the bridge. Around the same period, iron began to replace wood as the dominant structural material, later supplanted by steel. The division of the bridges into the four period groups shown in Table 3 plays a major role in the experiments. We refer to these groups as the *period groups*.

Table 3: Discretization of PERIOD property

Construction dates	PERIOD
1818–1870	Craftsman
1871–1890	Emerging
1891–1940	Mature
1941–present	Modern

An important requirement of learning is that it should reflect the role that the *PERIOD* property plays in the domain. More generally, learning should incrementally assimilate examples and reflect changes in the domain during learning. We now describe COBWEB, a system that attempts to perform this learning task.

3 COBWEB

COBWEB is a concept formation program for the creation of hierarchical classification trees (Fisher, 1987). COBWEB accepts a stream of examples described by a list of property-value pairs. Examples need not be classified as feasible, optimal, or by any other classification scheme. However, any *a priori* classification can be assigned to an example and treated as any other property.

A classification is 'good' if the description of an example can be guessed with high accuracy, given that it belongs to a specific class. COBWEB evaluates the classification of a set of examples into mutually-exclusive classes $C_1, C_2, \ldots, C_n$ by a statistical function called *category utility (CU)*:

$$CU = \frac{\sum_{k=1}^{n} P(C_k) \sum_i \sum_j P(A_i = V_{ij}|C_k)^2 - \sum_i \sum_j P(A_i = V_{ij})^2}{n} \tag{1}$$

where C_k is a class, $A_i = V_{ij}$ is a property-value pair, $P(x)$ is the probability of x, and n is the number of classes. The first term in the numerator measures the expected number of property-value pairs that can be guessed correctly by using the classification. The second term measures the same quantity *without* using the classes. Thus, the category utility measures the *increase* of property-value pairs that can be guessed *above* the guess based on frequency alone. The measurement is normalized with respect to the number of classes.

When a new example is introduced, COBWEB tries to accommodate it into an existing hierarchy starting at the root. The system performs one of the following operators (see (Fisher, 1987)) for a detailed description of these operators):

(1) expanding the root, if it does not have any sub-classes, by creating a new class and attaching the root and the new example as its sub-classes;
(2) adding the new example as a new sub-class of the root;
(3) adding the new example to one of the sub-classes of the root;
(4) merging the two best sub-classes and putting the new example into the merged sub-class; or
(5) splitting the best sub-class and again considering all the alternatives.

If the example has been assimilated into an existing sub-class, the process recurses with this class as the top of a new hierarchy. COBWEB again uses category utility to determine the next operator to apply.

Figure 3 shows the hierarchy generated by COBWEB after learning 60 examples ordered by their date of construction. The classes are described using their characteristic properties only. Characteristics are property values that satisfy $P(A_i = V_{ij}|C_k) \geq threshold$ and $P(C_k|A_i = V_{ij}) \geq threshold$, where *threshold* is a pre-determined fixed value[5]. The reference symbol of the class and the number of examples below each class (in parentheses) are also provided.

COBWEB predicts using a mechanism similar to the one used for augmenting the hierarchy by new,

[5]Potentially, this value can be learned for each domain. In the experiments described later, this value is fixed to 0.75. Other definitions of characteristic or normative values appear in (Fisher, 1987; Kolodner, 1983).

partially described examples but allowing only operator 3 to apply. COBWEB sorts the new example through the hierarchy to find the best host for the new example. Because only operator 3 is allowed, the best host is a leaf class (i.e., a previously learned example) that is used to complete the description of the new example. In terms of synthesis, this is a very conservative strategy, as it allows only the exact duplication of existing designs.

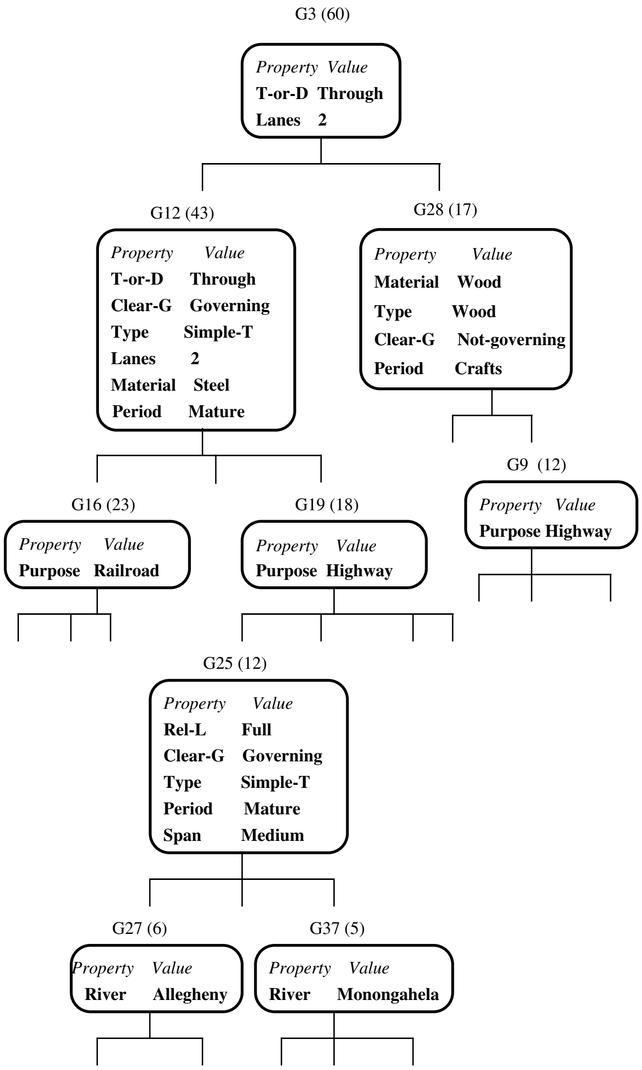


Figure 3: Design concept hierarchy after learning 60 examples.

An important issue in learning is the scaleability of the approach. Figure 4 illustrates the complexity of the classification hierarchy as a function of the number of training examples from the domain of Pittsburgh bridges. The average branching factor (solid line) of the hierarchy remains relatively constant around 2.8. This confirms Fisher's comment that COBWEB constrains itself to produce a small and useful number of classes without an *a priori* specification of an upper bound on the number of classes (Fisher, 1987). The size of the hierarchy (dashed line), normalized with respect to the number of examples, remains essentially constant, indicating that the hierarchy grows linearly with the number of examples.

Since all the examples are stored in the hierarchy as leaf classes, linear growth is the best possible and is a very encouraging result. However, when learning in a complex design domain, even such growth might constitute a problem. If this problem arises, it can be solved by an extension to COBWEB that constrains the size of the classification tree.

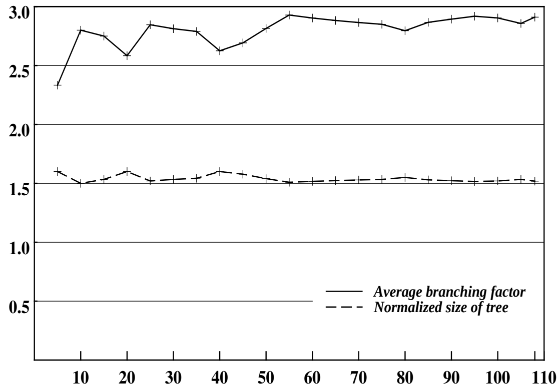


Figure 4: Size of classification hierarchy

To illustrate COBWEB's synthesis process, assume that COBWEB is required to synthesize a *highway* bridge on one of the rivers where *vertical clearance governs* using the knowledge shown in Figure 3. The following description is a high level interpretation of the prediction behavior of COBWEB. The system designs by classifying the given specification down the hierarchy. Since vertical clearance governs, the design description is refined using class G12, characterized by clearance governing. Class G19 is chosen next, since it represents *highway* bridges. At this stage synthesis should terminate and candidate designs (e.g., descendants of G19) should be returned, since all the known specifications have been met. However, the leaf prediction method used by COBWEB attempts to find a single leaf class. Thus, the synthesis process does not terminate at class G19. Rather, COBWEB continues to apply operator 3 with the guidance of *CU* until one leaf is selected as the candidate design.

COBWEB partially supports the requirements presented in Section 1. It allows a flexible use of the hierarchy; any specification, partial or complete, can be used to retrieve a candidate design. However, the system has a number of drawbacks for the use intended here, described below. Some of these drawbacks are later addressed in extensions of COBWEB.

(1) COBWEB can only handle nominal properties. CLASSIT is a descendant of COBWEB that can also handle continuous properties (Gennari et al., 1989). Elsewhere, we contrast its approach with the extension to continuous properties implemented in ECOBWEB (Reich, 1991) and conclude that ECOBWEB's strategy is more "natural" and flexible.
(2) COBWEB has a stiff prediction/design scheme. It makes a strong commitment to continue the design process until a complete existing design is retrieved. Generation of new designs or accommodation of subjective judgment is not allowed. In addition, leaf prediction is not adequate since it produces only one candidate design for a given specification. Leaf prediction is also susceptible to noise and may result in unnecessarily high error rates.
(3) COBWEB uses only the syntactic measure of category utility to guide its learning and prediction. No domain dependent or independent knowledge is used, although if available, this could enhance learning substantially.

Several tests are described in the following sections. First, COBWEB's ability to capture an evolving domain is tested and its ability to model few-to-many mapping is assessed. Then, extensions that address the first two issues described are tested. The third issue remains the subject of future research.

4 Testing of COBWEB

4.1 Testing procedures

There are varrious tests which can be used to assess learning programs. Each is suitable for different purpose (Kibler and Langley, 1988).

(a short summary)

For example, if one wishes to test the ability of a batch learning program to learn knowledge that transfers to new problems, one should prefer runing a cross validation test (Breiman et al., 1984). For example sets of over 100 examples, a good test is the *V-fold Cross Validation (V-CV)* experiment. In this test, the example set is divided into *V* subsets and *V* tests are conducted where all subsets but one are used for training and the remaining subset for testing.

(Stirling and Buntine, 1988)

Two type of tests are performed: performance and coverage.

Performance tests are most commonly used for assessing the performance of machine learning.

The test is rerun with random divisions into subsets to collect statistics. We call this test an $n \times$ *V-CV*, where *n* is the number of random divisions. The statistics are necessary to average the order effect in incremental learning programs.

Coverage test assesses whether the learning program can reproduce the *complete* set of examples by only observing the specification properties.

Most of the experiments reported in the following sections are coverage tests. Their procedure is now described in detail. As indicated in Section 2, the *PERIOD* of the bridges is an important property for classifying the bridges. This property plays a major role in the experiments described in the following sections.

The coverage experiments are performed as follows. COBWEB learns the 108 complete examples in increments of 5 and gradually creates a classification hierarchy. After each 5 examples, COBWEB uses the current hierarchy to find the best host for each of the examples in the entire set by using the specification properties only. The best host is used to perform the following tests:

(1) **The adaptation ability of COBWEB.** In this test, a match between the *PERIOD* property values of the example and the host is calculated. If the values are equal the match is 1, otherwise it is 0. A good match suggests that COBWEB is able to distinguish between the different period groups of bridges because the example bridge is designed by constructing its description from a host bridge of the same period.
(2) **The indexing ability of COBWEB.** This ability is calculated by the match between the specifications of the example and the host. The match is the number of equal specification property-values divided by the number of specification properties. This match measures the approximations introduced by the indexing scheme of COBWEB. It can be viewed as the

Table 4: Match between an example and its prediction

	Property	Input specification	Complete example	Predicted host	Match
Specification	RIVER	Monongahela	Monongahela	Monongahela	1
	LOCATION	3	3	4	0
	PERIOD	Craftsman	Craftsman	Craftsman	1
	PURPOSE	Highway	Highway	Railroad	0
	LANES	2	2	2	1
	LENGTH	Long	Long	Short	0
	CLEAR-G	Not-governing	Not-governing	Not-governing	1
Design Description	T-OR-D		Through	Through	1
	MATERIAL		Wood	Wood	1
	SPAN		Short	Medium	0
	REL-L		Small	Small	1
	TYPE		Wood	Wood	1

verification of the specifications.

(3) **The synthesis ability of COBWEB.** In this test, a match between the design descriptions of the example and the host is calculated similarly to the match of the specification properties. This match summarizes the ability of COBWEB to capture the information in the examples assimilated, to reconstruct them, and to extrapolate the knowledge to the rest of the example set.

An example of a match calculation is given in Table 4. The match between the *PERIOD* property values is 1, the match between the specification properties is $\frac{4}{7}$, and the match between the design description properties is $\frac{4}{5}$. The overall match in each of these three categories is the average of match values obtained from testing on the entire example set.

4.2 Results

This section describes the tests of the adaptation, indexing, and synthesis abilities of COBWEB. Although the adaptation and synthesis abilities are part of COBWEB, they have not been explicitly demonstrated in the past.

(1) **The adaptation ability of COBWEB.** Figure 5 describes the relative number of examples that are predicted using an example from the *same* period. The figure shows the matching for each period group separately. This demonstrates the performance of the algorithm better than matching for the complete example set. The results show that examples from the first period (*craftsman*) are always predicted using an example from the same period group. After learning 20 examples, of which 2 are from the second period (*emerging*), the algorithm is able to predict 66% of the examples of the second period group using these 2 examples. A similar pattern of rapid adaptation occurs for the other period groups as well. The drop in the *emerging* group between examples 90 to 108 can be attributed to the high correlation between this group and the *mature* group. This correlation causes the prediction of examples of the *emerging* group to rely more on the *mature* group as the relative number of the latter increases. This reliance, however, does not affect the two other matches (e.g., specifications and design descriptions to be described).

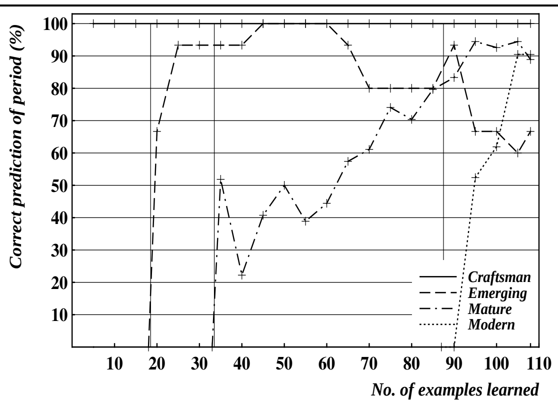


Figure 5: Predictions of period groups

The results demonstrate that COBWEB incrementally *adapts* to a changing domain exemplified by the four period groups. The ability to adapt to changing domains is a direct consequence of COBWEB's sensitivity to the order presentation of the examples. This order effect is further discussed in Section 5.1.

(2) **The indexing ability of COBWEB.** The results of matching the specification properties are shown in Figure 6a, and are contrasted to a simple frequency-based prediction scheme in Figure 6b. The frequency-based prediction is performed by taking the most frequent property-value pairs of the example set as the prediction. For each period group, the degree of matching increases as more examples of the group are augmented to the classification hierarchy. Note that transitions in the frequency match occur when one period group starts to dominate the entire set of examples.

The fact that there is never a 100% match of the specification means that COBWEB does not guarantee perfect indexing of previously stored examples. Nevertheless, the matching of the specifications provided by the hierarchy is better than the frequency-based matching. This observation holds in all later experiments as well.

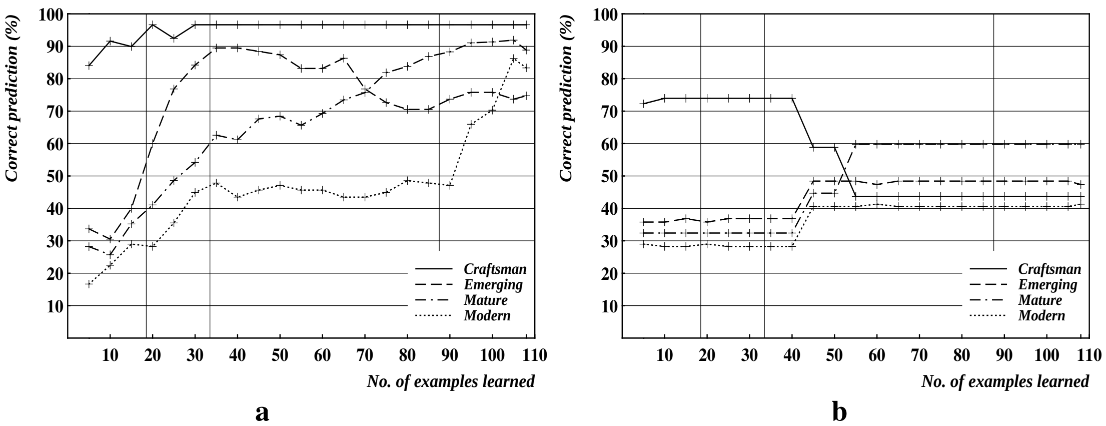


Figure 6: Predictions of specification properties

(3) **The synthesis ability of COBWEB.** The results of matching the design properties based on the specification properties are shown in Figure 7a, and are contrasted to a simple frequency-based prediction

scheme in Figure 7b. Examples from three of the periods (*craftsman, mature* and *modern*) are predicted better by the hierarchy as more examples of these groups are learned. Examples from the *emerging* period are predicted better when this group dominates the example set. However, for this group, the performance of the hierarchy toward the end of the learning process is not better than the frequency-based prediction, as seen in Figure 7b. The overall average performance of about 75% suggests that the example set is not sufficient to "cover" the domain. This accuracy level is an upper bound to the predictive accuracy in a performance test.

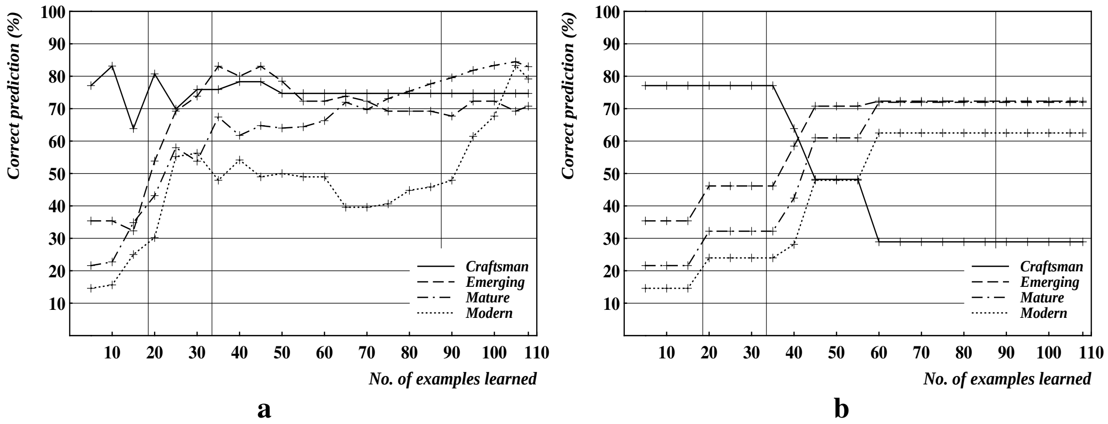


Figure 7: Predictions of design properties

5 ECOBWEB

ECOBWEB (Enhanced COBWEB) is a system that implements several extensions intended to broaden the applicability of COBWEB to design domains. These extensions and their tests are described in this section.

5.1 Extension to continuous values

Real artifacts are described by continuous as well as nominal properties. This holds true even for the simplified bridge design domain where *PERIOD* and *LENGTH* are continuous specification descriptors and *SPAN* and *REL-L* are continuous design descriptors. Also, the *LANES* descriptor is an integer specification that can be treated as continuous with additional rounding to the nearest integer.

The method of calculating the probabilities in Equation (1) is inadequate for continuous properties since the probability of a single event in a continuous distribution is zero. Since the objective is to maximize CU, it is best to combine property-values that correlate with similar classes. This results in a higher contribution to CU than the contribution of the separate $P(A_i = V_{ij})$ terms (Reich, 1991). For numerical properties this combination is naturally obtained by calculating the mean of the property-values in a class. This is the basis for the heuristic approach for the accommodation of continuous properties into the formulation of CU. The following transformation is used to replace the terms in Equation (1) for handling continuous properties. The first term of CU is transformed as follows:

$$\sum_j P(A_i = V_{ij}|C_k)^2 \quad \Rightarrow \quad P(A_i = \overline{V_i}|C_k)^2 \quad \Rightarrow \quad \left(\int_{-\frac{d_i}{\sigma_i}}^{+\frac{d_i}{\sigma_i}} p_i(x)dx \right)^2 \qquad (2)$$

where: $2d_i = \frac{\textit{expected range of property values of } A_i}{\textit{expected number of distinct intervals of property } A_i}$, $\overline{V_i}$ is the mean of the values of A_i in C_k, and σ_i is the standard deviation of the values of A_i in C_k. The second term of CU, $\sum_j P(A_i = V_{ij})^2$, is calculated similarly by calculating σ_i as the standard deviation of values of A_i in the root of the classification. The sensitivity of ECOBWEB's performance to the choice of $2d_i$ is low (Reich, 1991); a good domain-independent default value is $2d_i = 10$.

The probability distribution $p_i(x)$ must be selected before starting the learning process. The extension does not commit to any specific distribution. This is in contrast to several other systems which handle continuous properties (e.g., CLASSIT (Gennari et al., 1989) and AUTOCLASS (Cheeseman et al., 1988)) which always assume a normal distribution for continuous properties (compared to the equal distribution for nominal properties). Ideally, the selection should be based on some domain knowledge; alternatively, common probability distributions can be assumed. The default choice is the normal distribution:

$$p_i(x) = \frac{1}{\sigma_i\sqrt{2\pi}} e^{-(x-\overline{V_i})^2/2\sigma_i^2}. \qquad (3)$$

A distribution that proves to be deficient can be replaced without reprocessing of the previous training examples.

To test the extension, the original examples from the domain of Pittsburgh bridges were used. However, the two continuous design descriptor properties (i.e., *SPAN* and *REL-L*) were retained as discretized to allow for future comparisons with other learning approaches. The use of continuous properties in the design description would prevent most concept learning programs, including C4.5 to be discussed shortly, from predicting them, whereas ECOBWEB performs equally well with continuous design properties (Reich, 1991).

Table 5 provides the predictive accuracy of COBWEB, ECOBWEB, and the supervised concept learning program C4.5 (Quinlan et al., 1987) in a 10-fold cross validation performance test performed on the Pittsburgh bridge data. C4.5 is a descendent of ID3 (Quinlan, 1986) that implements various enhancements including pruning techniques, rule creation, etc. The last line in the table provides the predictive accuracy summed over the five design description properties, and its standard deviation in parentheses. Since the statistics of ECOBWEB and COBWEB's performance are the averages of 10 tests, the maximum and miminum accuracy levels are also provided for ECOBWEB.

Compared to ECOBWEB's learning with combined continuous and nominal properties, COBWEB's performance with discretized properties, summed over the design description properties, is inferior with a significance beyond the 0.01 level. C4.5 performs better than the average performance of ECOBWEB,

Table 5: Prediction accuracy of ECOBWEB, COBWEB, and C4.5 on Pittsburgh bridges data

program	C4.5	ECOBWEB			COBWEB
test	10-CV	10×10-CV			10×10-CV
properties	mixed	mixed			nominal
		average	max	min	
T-or-D	85	85.3	85.3	85.3	85.3
Material	85	81.6	87.7	73.3	75.8
Span	68	57.3	65.2	50.0	60.1
Rel-L	68	61.3	70.0	54.3	58.3
Type	56	52.4	59.0	49.5	43.5
Summary	—	67.61 (1.70)	—	—	64.62 (1.75)

but is inferior, to the *best* (i.e., the max column) performance acheived by ECOBWEB. Therefore, for this batch task C4.5 is better on the average, but ECOBWEB is sometimes better. The reason that ECOBWEB is sometimes better is because it encounters a *favorable* ordering of examples that it can exploit, in contrast to C4.5 which cannot.

An important research topic is the generation of "good" orderings of examples that can result in the best possible performance of ECOBWEB. This, however, can be utilized only in two ways: (1) all the examples are available and (2) the learning selects its own training examples from the teacher. In the first case, the availablity of examples eliminates the need for incremental learning, it will inhance ECOBWEB's ability to compete with batch programs. In the second case, the ability to ask for good meaningful training examples (i.e., having an *experimentation* capability) can lead to significantly improved performance.

Another enhancement for ECOBWEB is the addition of a noise handling mechanism, similar to the one developed for COBWEB (Fisher, 1989). This will enhance ECOBWEB's performance and will result in a fairer comparison with C4.5 which has such mechanism.

Independent of whether ECOBWEB's performance can be enhanced on the 10-fold cross validation task, it must be rememberred that C4.5 serves only as a calibration for ECOBWEB and not as a test whether ECOBWEB's approach is appropriate. First, in the test, only one candidate is generated for each specification instead of several. Second, two design description properties were discretized to allow for C4.5 to operate on them. Third, the designs generated by ECOBWEB are always compatible, being existing designs, rather than ad-hoc, even if probable, collection of properties. Fourth, the test is a batch, rather than incremental, test.

Figure 8 shows the results of the coverage experiment. The prediction of specification properties, shown in Figure 8a, is slightly better, on the average, than that presented in Figure 6a for nominal properties. The prediction of design description properties, shown in Figure 8b, is considerably better than that presented in Figure 7a. The quality of the classification tree is better than the original hierarchy, resulting in similar, yet more refined, meaningful classes and the same tree complexity.

The enhanced performance in both performance and coverage tests suggests that the new approach for handling the continuous property values is more natural than the use of subjectively imposed ranges. Imposed ranges can cause a learning algorithm to treat examples that differ by only a slight amount in one continuous property differently, classifying these examples into different classes.

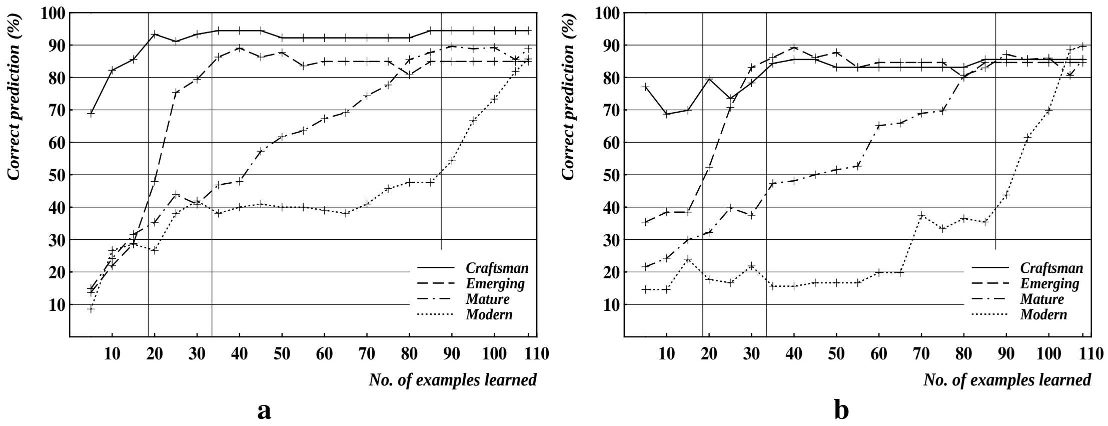


Figure 8: Predictions with continuous specification properties

To calibrate the performance of COBWEB versus C4.5 another test was performed. In this test a separate classification tree is generated for each design property. This decomposes the few-to-many mapping into a set of many-to-one mappings. The results show an increase in predictive power at the cost of using more extensive memory and computations (See Figure 9 compared to Figure 7a). Note that as discussed before, this is not the task for which ECOBWEB was designed.

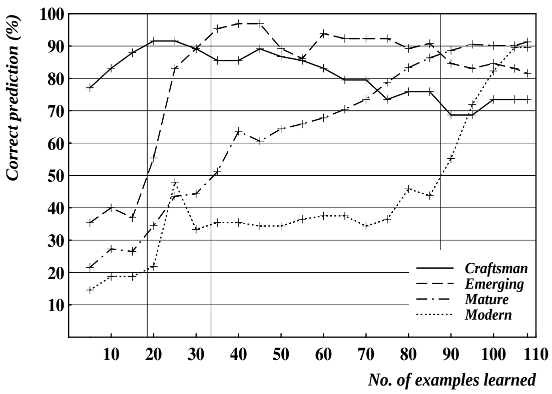


Figure 9: Predictions of design properties with multiple trees

As shown in Table 5, C4.5 performs better than COBWEB or ECOBWEB in this domain because it uses separate trees for each design property. When used in this way, COBWEB or ECOBWEB also perform better on this domain. As discussed before, a drawback of using separate trees is that they hide the real structure of the artifacts and lose the correlation information between the various design description properties.

These results, by themselves, do not support the claim that unsupervised concept learning is *required* for capturing synthesis knowledge. The proof of this conjecture will emerge when more realistic design domains are tested.

5.2 New prediction methods

ECOBWEB can use several synthesis methods. This section illustrates them using Figure 3 again[6]. In the same design problem as before, i.e., to design a *highway* bridge at a location where *clearance governs*, ECOBWEB starts designing by classifying the new specification with the hierarchy. The top class of ECOBWEB's knowledge represents *through* bridges with *2* lanes. These property-values are assigned to the new design: the first property is a design refinement and the second property is a specification elaboration. Since vertical clearance governs, the design description is again refined using class G12. The new design will be a *simple-truss* made of *steel*. Class G19 is again chosen next, since it matches the given *highway* specification. At this stage the design terminates, since all the specifications have been met, but the design is only partially specified as the main span and the relative length of the bridge have not been determined. Since the specification of the new bridge is abstract, it is expected that such an abstract design would be obtained,

ECOBWEB can use refinement strategies to complete the design or retain the abstract design as the solution. For example, ECOBWEB can deliver the set of 18 bridges under class G19 as design candidates in a pure *case-based* approach; it can generate a new prototype from the most frequent property-value pairs in G19 and deliver it with a set of possible variations as in a pure *prototype-based* design; or it can deliver a large set of candidates generated from the combination of the property-value pairs of the 18 designs, a different form of prototype-based design. It can be observed that the path traversed by the guidance of the category utility function, *CU*, can be interpreted as a progressive matching of the specifications or even as a design derivation[7]. This is desirable, although the coherence of the knowledge structure generated is not conceptualized as a criterion for the success of the learning approach.

ECOBWEB's synthesis methods can be described along two dimensions: the refinement process which can be *extensional* or *intentional*; and the generation process which can be *case-based* or *prototype-based*. Figure 10 illustrates these dimensions. In the extensional approach, refinement classifies the design with a new subclass starting from the top class (class 1 in Figure 10) until the process terminates (class 3). In this view, a class represents the extension of all its leaves. In the intentional approach, while classifying the new problem, characteristic property-values of the classes traversed (classes 1, 2, and 3 in Figure 10) are assigned to the new design as described in the example before.

The intentional mechanism can overcome contradictions in the hierarchy such as the one seen in Figure 11 which describes the classification hierarchy generated by COBWEB after training with 108 examples from the Pittsburgh bridge domain. In the absence of characteristic values, dominant property-value pairs are displayed for several classes (ordinary font below a horizontal line). If a *railway* (RR) bridge is to be designed, COBWEB will try to fit it into G16. Sorting it further down might result in choosing G24 as the host, which is characterized as a *highway* bridge. In this case, the incorrect match concerns a specification property; however, it might be present in the case of design-description properties as well.

The intentional case-based prediction method was tested in a coverage test. In the tests, only one design is synthesized for each specification. This is done by continuing applying the learning operator 3, guided by CU, until a leaf node is reached or all the design description properties have been assigned by the intentional mechanism. The results show that the intentional prediction scheme, shown in Figure

[6]The hierarchy generated by ECOBWEB while using continuous properties is slightly different.

[7]It is important to acknowledge that the sequence of design description property-value assignments does not approximate in any way the explicit intent and domain knowledge on the order in which design derivations are to proceed. Such concerns may be supported when domain knowledge is incorporated in the learning or synthesis processes.

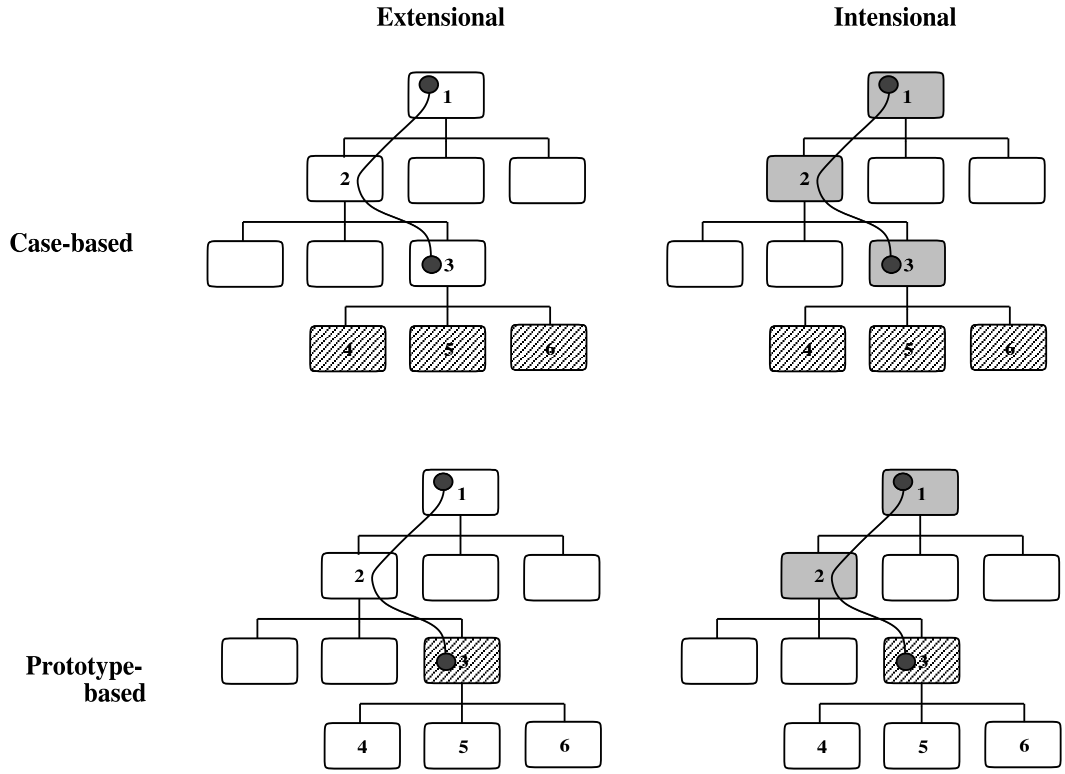


Figure 10: Synthesis methods

12, performs better than the original scheme (Figure 7a); performance experiments in additional domains (Reich, 1991) confirm this statement.

6 Summary

This paper discusses some of the issues in learning synthesis knowledge. A system that learns synthesis knowledge should: (1) be incremental and have the ability to track a changing domain; (2) support the creation of few-to-many mappings; (3) allow the use of continuous as well as nominal property types; and (4) support the prediction of several outcomes for a given specification. COBWEB, a concept formation program is proposed as the basic mechanism for acquiring synthesis knowledge from design examples. This paper has demonstrated that COBWEB effectively satisfies the first two requirements for synthesis. The two additional requirements, not supported by COBWEB, are addressed by the development and implementation of extensions embedded in a descendant system called ECOBWEB.

The tests of ECOBWEB's performance presented in the paper include statistical performance tests and coverage tests; the latter emphasizing the process of incremental assimilation of knowledge from examples. These tests demonstrate ECOBWEB's potential for deriving synthesis knowledge from existing designs.

ECOBWEB is limited along several dimensions, which remain the subject of future research. First, it should handle structured descriptions of artifacts, as manifested in many design domains. This can be

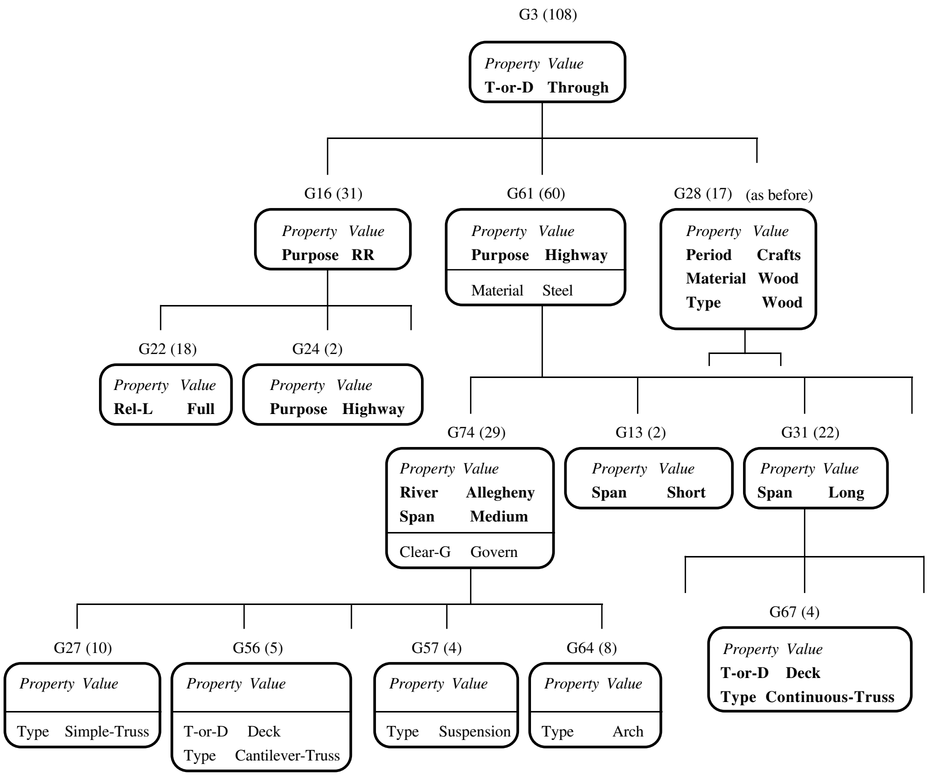


Figure 11: Design concept hierarchy after learning 108 examples

achieved by incorporating the principles from another descendant of COBWEB (Thompson and Langley, 1991). Second, the exploitation of mechanisms for ordering examples and for handling noise will further enhance ECOBWEB' performance. Third, in some design domains, there exists some synthesis knowledge, or it can be derived by means other than learning (e.g., from interviewing expert designers), which can potentially enhance the learning performance. Currently, ECOBWEB cannot use pre-existing knowledge in its learning and prediction and, therefore, should be modified to take this knowledge into account. Initial studies along these lines appear in (Reich, 1991).

Acknowledgments

This work has supported in part by the Engineering Design Research Center, a National Science Foundation Engineering Research Center, and the Sun Company Grant for Engineering Design Research. The performance of C4.5 on the Pittsburgh bridge database was obtained from Ross Quinlan.

References

Araya, A. A. and Mittal, S. (1987). Compiling design plans from descriptions of artifacts and poblem solving heuristics. In *Proceedings of The Tenth International Joint Conference on Artificial Intelligence*, pages 552–557, Milan, Italy. Morgan Kaufmann.

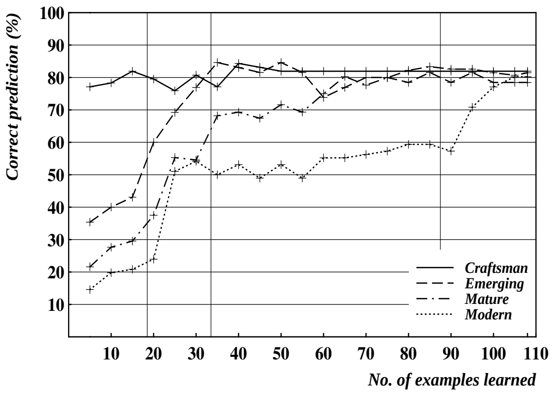


Figure 12: Predictions of design properties with characteristics prediction

Arciszewski, T., Mustafa, M., and Ziarko, W. (1987). A methodology of design knowledge acquisition for use in learning expert systems. *International Journal of Man-Machine Studies*, 27(1):23–32.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Belmont, Waldsworth, CA.

Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., and Freeman, D. (1988). AutoClass: A Bayesian classification system. In Laird, J., editor, *Proceedings of the Fifth International Conference on Machine Learning*, pages 54–64, Ann Arbor, MI. Morgan Kaufmann.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(7):139–172.

Fisher, D. H. (1989). Noise-tolerant conceptual clustering. In *Proceedings of The Eleventh International Joint Conference on Artificial Intelligence, Detroit, MI*, pages 825–830, San Mateo, CA. Morgan Kaufmann.

Fisher, D. H. J., Pazzani, M. J., and Langley, P., editors (1991). *Concept Formation: Knowledge and Experience in Unsupervised Learning*. Morgan Kaufmann, San Mateo, CA.

Gennari, J. H., Langley, P., and Fisher, D. (1989). Models of incremental concept formation. *Artificial Intelligence*, 40(1-3):11–61.

Kibler, D. and Langley, P. (1988). Machine learning as an experimental science. In Sleeman, D., editor, *Procedings of the Third European Working Session on Learning*, pages 81–92, Aberdeen. Pitman.

Kolodner, J. L. (1983). Maintaining organization in a dynamic long-term memory. *Cognitive Science*, 7:243–280.

Lu, S. C.-Y. and Chen, K. (1987). A machine learning approach to the automatic synthesis of mechanistic knowledge for engineering decision-making. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, 1(2):109–118.

Mackenzie, C. A. and Gero, J. S. (1987). Learning design rules from decisions and performances. *Artificial Intelligence in Engineering*, 2(1):2–10.

Mitchell, T., Mahadevan, S., and Steinberg, L. (1985). LEAP: A learning apprentice for VLSI design. In *Proceedings of The Ninth International Joint Conference on Artificial Intelligence, Los Angeles, CA*, pages 573–580, San Mateo, CA. Morgan Kaufmann.

Mostow, J. (1989). Design by derivational analogy: Issues in the automated replay of design plans. *Artificial Intelligence*, 40(1-3):119–184.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

Quinlan, J. R., Compton, P., Horn, K. A., and Lazarus, L. (1987). Inductive knowledge acquisition: A case study. In Quinlan, J. R., editor, *Applications of Expert Systems*. Addison-Wesley, Reading, MA.

Reich, Y. (1991). *Building and Improving Design Systems: A Machine Learning Approach*. PhD thesis, Department of Civil Engineering, Carnegie Mellon University, Pittsburgh, PA. (Available as Technical Report EDRC 02-16-91).

Reich, Y. and Fenves, S. J. (1991). The formation and use of abstract concepts in design. In Fisher, D. H. J., Pazzani, M. J., and Langley, P., editors, *Concept Formation: Knowledge and Experience in Unsupervised Learning*, pages 323–353, Los Altos, CA. Morgan Kaufmann.

Schlimmer, J. C. and Granger, R. H. J. (1986). Beyond incremental processing: tracking concept drift. In *Proceedings of AAAI-86*, pages 502–507, Philadelphia, PA. Morgan Kaufmann.

Stirling, D. and Buntine, W. (1988). Process routings in a steel mill: a challenging induction problem. In Gero, J. S. and Stanton, R., editors, *Artificial Intelligence Developments and Applications*, pages 301–313. North-Holland, Amsterdam.

Thompson, K. and Langley, P. (1991). Managing components in object concept formation. In Fisher, D. and Pazzani, M., editors, *Concept Formation: Knowledge and Experience in Unsupervised Learning*, Los Altos, CA. Morgan Kaufmann.

Whitehall, B. L., Lu, S. C.-Y., and Stepp, R. (1990). CAQ: A machine learning tool for engineering. *Artificial Intelligence in Engineering*, 5(4):189–198.

Table 1: Examples of bridge descriptions

	property	Example 1	Example 2	Example 3
specification	NAME[1]	16th St. B.	Fort Duquesne B.	Penn. Turnpike B.
	IDENTIFIER[1]	E19	E84	E88
	RIVER	Allegheny	Allegheny	Allegheny
	LOCATION	29	24	37
	PERIOD	Craft	Modern	Modern
	PURPOSE	Highway	Highway	Highway
	LANES	2	6	4
	LENGTH	Medium	Short	Long
	CLEAR-G	Not-governing	Governing	Not-governing
design description	T-OR-D	Through	Through	Deck
	MATERIAL	Wood	Steel	Steel
	SPAN	Medium	Medium	Long
	REL-L	Small	Full	Full
	TYPE	Wood	Arch	Continuous-Truss

[1]These properties are not used in the learning process.

Table 2: Summary of property-values

Specification

RIVER	108	**LOCATION**	107	**PERIOD**	108	**PURPOSE**	108
Youghiogheny	3	Varies from 1 to 52		Craft	18	Highway	71
Ohio	15			Emerging	15	RR	32
Monongahela	41			Mature	54	Aqueduct	4
Allegheny	49			Modern	21	Pedestrian	1

LANES	92	**LENGTH**	81	**CLEAR-G**	106
1	4	Short	12	G	80
2	61	Medium	48	N	26
4	23	Long	21		
6	4				

Design Description

T-OR-D	102	**MATERIAL**	106	**SPAN**	92	**REL-L**	103	**TYPE**	105
Through	87	Steel	79	Long	30	Full	58	Continuous-Truss	10
Deck	15	Iron	11	Medium	53	Medium	15	Cantilever-Truss	11
		Wood	16	Short	9	Small	30	Arch	13
								Simple-Truss	44
								Suspension	11
								Wood	16

Table 3: Discretization of PERIOD property

Construction dates	PERIOD
1818–1870	Craftsman
1871–1890	Emerging
1891–1940	Mature
1941–present	Modern

Table 4: Match between an example and its prediction

	Property	**Input specification**	**Complete example**	**Predicted host**	**Match**
Specification	RIVER	Monongahela	Monongahela	Monongahela	1
	LOCATION	3	3	4	0
	PERIOD	Craftsman	Craftsman	Craftsman	1
	PURPOSE	Highway	Highway	Railroad	0
	LANES	2	2	2	1
	LENGTH	Long	Long	Short	0
	CLEAR-G	Not-governing	Not-governing	Not-governing	1
Design Description	T-OR-D		Through	Through	1
	MATERIAL		Wood	Wood	1
	SPAN		Short	Medium	0
	REL-L		Small	Small	1
	TYPE		Wood	Wood	1

Table 5: Prediction accuracy of ECOBWEB, COBWEB, and C4.5 on Pittsburgh bridges data

program	C4.5	ECOBWEB			COBWEB
test	10-CV	10×10-CV			10×10-CV
properties	mixed	mixed			nominal
		average	max	min	
T-or-D	85	85.3	85.3	85.3	85.3
Material	85	81.6	87.7	73.3	75.8
Span	68	57.3	65.2	50.0	60.1
Rel-L	68	61.3	70.0	54.3	58.3
Type	56	52.4	59.0	49.5	43.5
Summary	—	67.61 (1.70)	—	—	64.62 (1.75)

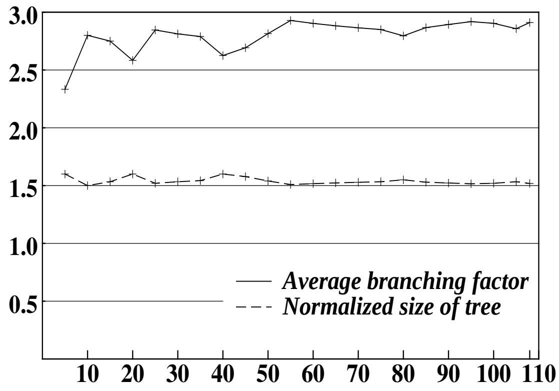


Figure 4: Size of classification hierarchy

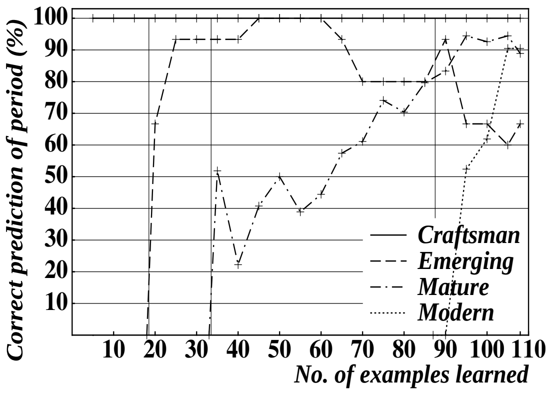


Figure 5: Predictions of period groups

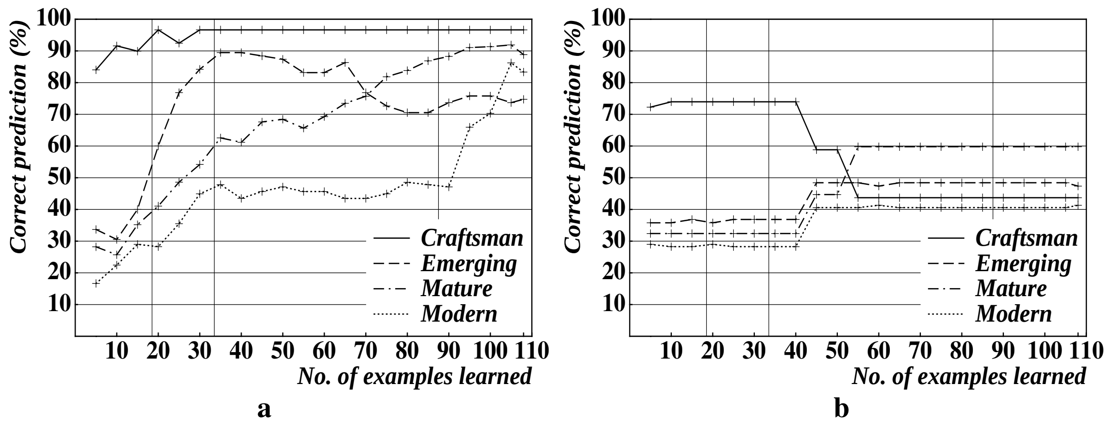
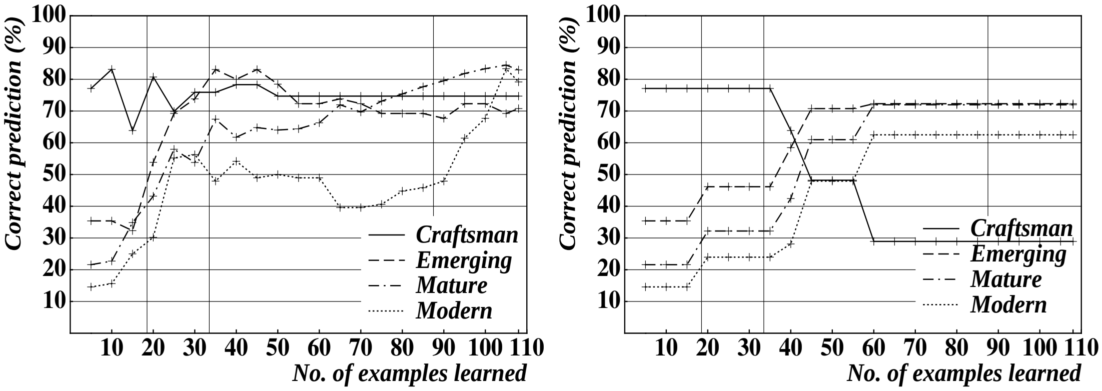
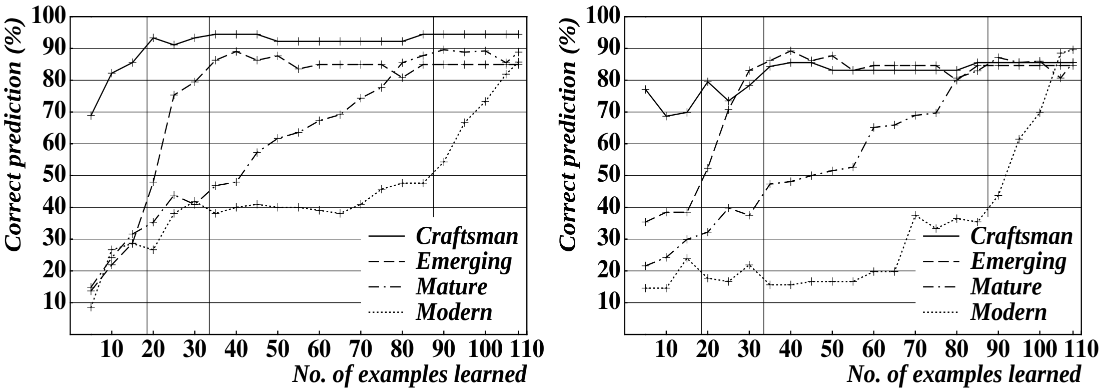


Figure 6: Predictions of specification properties



a **b**

Figure 7: Predictions of design properties



a **b**

Figure 8: Predictions with continuous specification properties

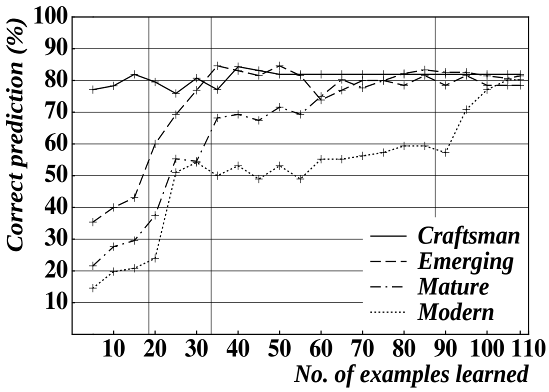


Figure 10: Predictions of design properties with intensional prediction