Automatic selection of examples for training a learning design system

Y. Reich

EDRC 12-42-91

Automatic selection or generation of examples for training a learning design system

Yoram Reich
Engineering Design Research Center
Carnegie Mellon University
Pittsburgh, PA 15213

September 25, 1991

Abstract

It has been observed that a careful selection of training examples improves the performance of supervised concept learning systems. This observation is important when examples are not available and are expensive to generate, or when using incremental learning systems. These two reasons are manifested in the context of BRIDGER, a system that learns to design cable-stayed bridges. This paper describes example selection techniques for the concept formation system ECOBWEB—the system that is used to acquire the synthesis knowledge in BRIDGER. The approach was implemented and tested. Preliminary results show that small improvements in syi uesis performance can be obtained by using the new technique.

1 Introduction

Machine learning techniques are emerging as important tools for. extracting knowledge from either existing designs or simulation of behavior of designs (Arciszewski, 1991; Buchanan et al., 1988; Lu and Chen, 1987; Reich, 1991). The availability of good sources of examples is crucial for obtaining a satisfactory performance of machine learning techniques. In many domains examples do not exist, but can be generated by running time-consuming simulation programs or by designing examples. This is the case for BRIDGER, a system that learns to design cable-stayed bridges. There are relatively few examples of cable-stayed bridges available and the generation of new examples is time consuming; it involves the selection of *good' specification and the synthesis, analysis and redesign of solutions to the selected specification. It is therefore important to devise a method that will allow the generation of the most useful examples, such that the improvement in knowledge due to learning is maximized at the expense of little resources spent for training.

The problem of selecting good examples is especially dominant when using an incremental learning system. In this case, even if all the examples are available, it is useful to impose a good ordering on the training examples. While using COBWEB (Fisher, 1987) in several design domains (Reich, 1991)

it has been observed that the order of training examples used to generate synthesis knowledge had a substantial effect on the synthesis ability. Deviations of 90% in synthesis performance were observed in statistical experiments with random orderings of training examples drawn from various domains. It may seem that if one orders examples to improve performance by analyzing all the examples, one defeats the advantage of the incremental nature. However, if all the data is available, there is no reason not to take a full advantage of it.

In the development of COBWEB, the order effect on learning performance was partially addressed by the incorporation of two learning operators, split and merge, that *simulate* backtracking. These operators can perform relatively major modifications to the knowledge generated previously and therefore allow COBWEB to partially overcome from bad orderings of examples. Statistics of operator applications show that these two operators are usually used 10-15% out of the total number of operator applications. Therefore, the operators seem to reduce order effects, but according to the variations in synthesis performance described before, do not eliminate them.

Two important questions arise in relation to the order effect. The first question is whether order effects should be *completely* eliminated, loosing the ability to detect knowledge change? The answer is negative. Since design domains evolve, the detection of changes in knowledge is valuable. The second question is how can we avoid *bad* orderings of examples and exploit the incremental nature of concept learning systems to enhance their behavior? This may be achieved by a method that actively selects or orders examples. In the machine learning terminology, an acti\e selection of examples is also called *experimentation.* The second question is the focus of this study.

Machine learning research provides some insight into the effects of order of training examples on learning. Amsterdam, 1988 has shown that in a simple case of concept learning, adaptive example selection is more powerful than receiving a stream of examples from a teacher[1]. This suggests that there is merit to example selection as a solution to the second q lestion. Furthermore, concept learning systems have benefited from experimentations for some time (Mitchell et al., 1983; Gross, 1988). In addition, it has been observed that complex procedures can be learned by using a good sequencing of lessons (VanLehn, 1987).

More broadly, the performance of a learning system depends on its knowledge (learning operators and background information), bias[2] (concept description language) and its input.[3] A change in any of these aspects modifies the system performance. In cases of weak bias and knowledge, proper sequencing of examples can not only increase efficiency, but can also determine leamability.

This paper explores the way by which examples can be ordered or actively generated/selected to enhance the performance of ECOBWEB — a concept formation system for synthesis knowledge

[1]A slightly contradicting result on the type of example selection needed is reported by Ruff and Dietterich, 1988, who have shown that for a simple domain, the utility of simple selection scheme is higher than that of random observations and matches that of a careful example selection.

[2]The distinction between knowledge and bias can often be eliminated (Russell and Grosof, 1987).

[3]This roughly corresponds to (VanLehn, 1987) categories of inductive learners: the partially blind, the strongly prejudiced and the felicitiosly taught.

acquisition that extends the original COBWEB along several dimensions. We describe several heuristic approaches for **example** selection and contrast their behavior with previous experiments in the domain of Pittsburgh **bridges** (Reich and Fenves, 1991). We also test the approaches on several artificial domains. The comparison shows that ECOBWEB slightly improves its performance when it uses the new ordering methods.

The remainder of the paper is organized as follows. Section 2 briefly reviews BRIDGER, it focuses on the aspect of synthesis knowledge acquisition which is performed by ECOBWEB. Section 3 discusses the issue of order effect and some potential remedies for reducing it. Section 4 describes the method ECOBWEB uses for example selection. Section 5 provides two examples of using the new ordering schemes. Sections 6 and 7 describe possible extensions and conclude the study.

2 A brief overview of BRIDGER and ECOBWEB

BRIDGER is a system that assists in the preliminary design of cable-stayed bridges. Its architecture is based on a sequential view of design composed of problem analysis, synthesis, design analysis, redesign, and evaluation (Reich, 1990). The system is intended to provide computational support for all except the first task.

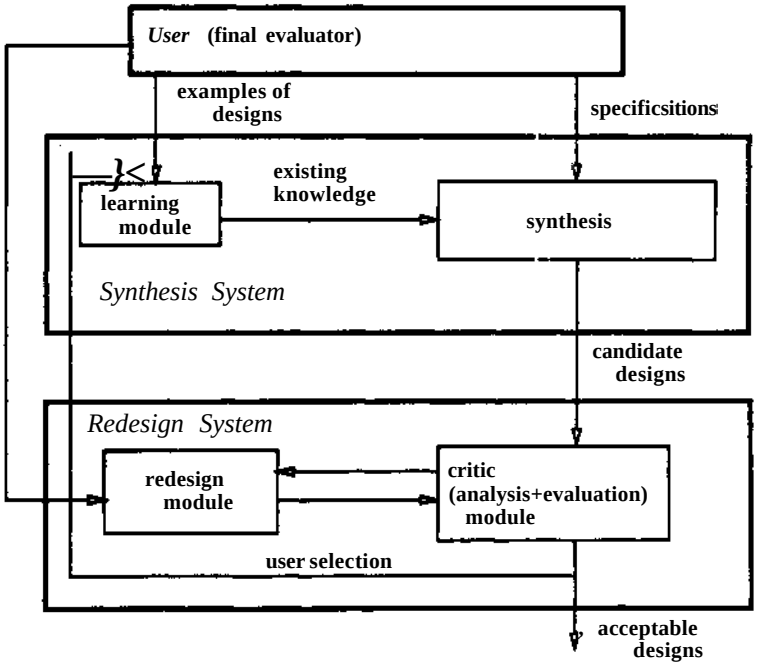

Figure 1: An overview of BRIDGER's architecture

BRIDGER contains two main subsystems: synthesis and redesign. The synthesis system is responsible for synthesizing several candidates from a given specification. It also augments its knowledge by successful design examples that are selected by the user. Candidate designs are submitted to the redesign

system for analysis and redesign. These processes iterates until acceptable designs are generated. The user can evaluate the solutions and select a subset for further training the synthesis system.

The synthesis system employs concept formation to incrementally create a hierarchical classification knowledge structure from the examples. Concept formation ability is believed to be fundamental to design (Reich and Fenves, 1991). The more examples BRIDGER incorporates, the better the quality of the knowledge generated. The classification hierarchy generated is used in synthesis of new designs. BRIDGER'S synthesis uses an enhanced version of COBWEB (Fisher, 1987), called ECOBWEB, that allows the use of continuous, ordinal, or nominal property types in bridge descriptions; and that can forget examples and correct the hierarchical knowledge organization based on some evaluation criteria (Reich and Fenves, 1991; Reich, 1991).

ECOBWEB (as well as COBWEB) builds the classification hierarchy in the following way. When a new design is introduced, ECOBWEB tries to accommodate it into the existing hierarchy starting from its top class. ECOBWEB can perform one of the following 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 design as its sub-classes;
(2) adding the new design as a new sub-class of the root;
(3) adding the new design to one of the sub-classes of the root;
(4) merging the two best sub-classes and putting the new design into the merged sub-class; or
(5) splitting the best sub-class and considering again all the alternatives.

If the design has been accommodated into an existing sub-class, the process recurses with this class as the top of a new hierarchy. ECOBWEB uses an evaluation function, called *category utility (CU)*, to determine the next operator to apply. The operator that its execution maximizes the utility function is selected. Category utility is calculated as follows:

$$CU = \frac{\sum_{i=1}^{n} P\{C_k\} \sum_{j} P\{A_i = V_i \wedge C_k\}^2 - \sum_{i} \sum_{j} P\{A_i = V_i\}^2}{n} \quad (1)$$

where: C* is a class, $A_i = V_y$ 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.

ECOBWEB synthesizes using a mechanism similar to the one used for augmenting the hierarchy by new designs, but allowing only operator (3) to apply. The synthesis mechanism also extends the original prediction methods of COBWEB (Reich, 1991).

3 Order effects and their elimination

Experimental studies in concept learning have used various example selection schemes to enhance their learning performance. Most of the strategies are based on focusing on the part of the example-space that is uncertain with respect to the concepts being learned (Mitchell et al., 1983; Gross, 1988). These methods rely completely on the labeling (e.g., positive or negative instances of the concept) of examples to determine the best area in the example-space to explore next. However, in concept formation, which is unsupervised, there are no *a priori* concept classes that serve as boundaries for focusing. Furthermore, concept classes are dynamically formed, modified and even discarded. Therefore, the focusing technique is inappropriate for our problem.

Ignoring the details of the *focusing' methods, they all use a metric that evaluates the current status of knowledge for determining the next step. If such a metric existed for a concept formation system, it could have been used as the basis for example selection.

Another approach for using a simple form of example selection is exemplified by ISG (Wisniewski et al., 1987). ISG creates a hierarchy of equivalence classes of situations (i.e., examples) that is used to generate diagnostic rules. ISG uses the ordering of properties and a perfect domain theory in the form of linear linkage rules (inverse of diagnostic rules) to generate one equivalence class at a time. ISG selects its own examples such that they do not belong to any of the existing equivalence classes. This approach will not work in general, and in particular not in the target domains of BRIDGER where such linkage rules do not exist.

4 ECOBWEB ordering scheme

Since the category utility function *(CU)* serves as a measure of the increase of property-value pairs that can be predicted using the classification, it can serve as a metric for evaluating the quality of the knowledge structure. As discussed previously, such a measure can be used for guiding ECOBWEB through a search in the space of possible knowledge structures by trying to maximize *CU* while learning additional examples. In general, this search yields a combinatorial explosion since the next training example can be constructed from any combination of property-value pairs. Therefore, such an approach requires heuristics for its practical application.

One such heuristic suggests that only small increments in knowledge are allowed. Small increments arise if the experiments generate examples that are relatively similar to previous examples. In ECOBWEB this is easily done by selecting examples that only vary slightly from the most frequent property-value pairs in the domain. In ECOBWEB these values are readily available at the root of the classification hierarchy. The collection of the most frequent property-value pairs is called *the frequent example* (FE).

There are several ways to design an example selection/generation technique for ECOBWEB. These ways differ in the amount of information they use. Three classes of techniques are discussed below.

Program independent/slight example-set dependent strategy (PISED).
This technique does not rely on any feature of ECOBWEB; therefore, it is independent of the program

that is learning from the data. It only makes a small use of the examples that have already been learned. The only information that is used is the one summarized in FE. Two variants on the approach exist.

- *Most-frequent technique.* This approach is only applicable for an ordering scheme, not for example creation. In this approach, the next example that is selected from the training set is the one that is most similar to FE. If this approach would be used for example creation, all the examples would be the same.
- *Most-distant technique.* This approach applies to both ordering end experimentation. In the *ordering* scheme, an example that is most distant from FE is selected from the training set. In the *experimentation* an example that is most distant from FE is generated from the domain. This approach relies on a heuristic opposite to the one described before; its application results in exploring aspects of the domain that were not covered by previous examples.

Program dependent/Average example-set dependent strategy (PDAED).
This technique makes use of the program mechanisms and significant aspects of the data for selecting/creating the next example. In the context of ECOBWEB, the techniques makes use of *CU* as an evaluation function for classifications. It attempts to find an example that when learned, will maximize the *CU* value of the top classification. The dependency on the example set is determined by the use of the top level of ECOBWEB'S hierarchy to make the decision. This level summarizes a rough classification of the example set. There are several ways to maximize *CU*\ for example by the use of gradient methods.

- *Univariate.* In this approach, FE is calculated first. Then, for each of the properties/?, n_p examples are generated where n_p is the number of possible values that property/? has. Each of the examples is generated by replacing the value of *p* in FE by the *riff* possible value. The category utility function of the top classification is calculated as if this example was a training example. This process is done for all the properties.

  At the end, the example that yielded the highest *CU* score is selected as the preferable new example. This constitutes a simple hill-climb control. In an example selection problem, the actual new training example is an existing example that is closest to the preferable example and in an example generation problem, the new training example is the preferable example.

  This calculation requires knowledge about the possible values of the nominal properties. The more values the program is familiar with, the richer the possible set of calculation performed.

- *Steepest.* This approach follows the univariate approach except that the preferable example is an example that each of its property-value pairs is replaced by the property-value pair that resulted in the highest *CU* score. Although the new example is completely different than FE, the approach still uses FE to calculate the *CU* values from which the new example is calculated. Therefore it also subscribes to the first heuristic discussed before.

This approach is very simple and general. It uses a very simple metric - *CU* - that only evaluates the clustering quality of the top level of the hierarchy. It follows a very simple heuristic for restricting the

search for the **new** example, i.e., calculating the gradient about the frequent example (FE). As a result, the complexity of the process is $O(m^2 n^2)$, where m is the number of properties describing examples and n is the maximum number of values for a nominal property.

Major Program dependent/Major example-set dependent strategy (PDMED).
This technique uses the complete structure of the knowledge that ECOBWEB generates. It requires an evaluation function that is based on the complete hierarchy. For example, knowledge (or hierarchy) utility is a measure that is derived by applying *CU* recursively starting from the root of the hierarchy (Reich, 1991):

```
knowledge - utility (class)                                          (2)
    if class is a leaf class
        return 0.0
    else
        return CUxn+ ]P_ P(Ck) x knowledge-utility (son).
                     kGsonsof class
```

After the calculation, the value is normalized by the number of properties describing artifacts.

5 Test cases

Two experiments were performed to assess the potential of the iew experimentation techniques. The first experiment tests the technique on a set of artificial domains and the second on the domain of Pittsburgh bridges. Artificial domains can help calibrate a learning method against known data, and real domains test a technique on data that is imperfect and noisy; both experiments are beneficial. In addition, the artificial domains test the experimentation aspect because examples that do not exist *a priori* are generated; whereas the real domain tests the ordering aspect of the technique because the technique is used to select examples from a pre-existing example set. In both experiments, the performance of ECOBWEB with with either the most-distant or the steepest gradient techniques was compared to the performance with random ordering of examples.

5.1 Artificial domains

The description of the three domains is given in Figure 2. They are represented as finite-state machines that can generate strings. These domains are similar to the domains appearing in (Fisher, 1987). Tests were performed with these and other artificial domains. In the tests, random example selection was compared to the most-distant and the steepest-gradient selection techniques. Since it is a selection task, the actual new training example is generated from the class that best matches the example selected by the strategy.

The metric for evaluating the experimentation scheme is the values of *CU* obtained in the learning process, the number of split and merge operators performed during training, and the prediction

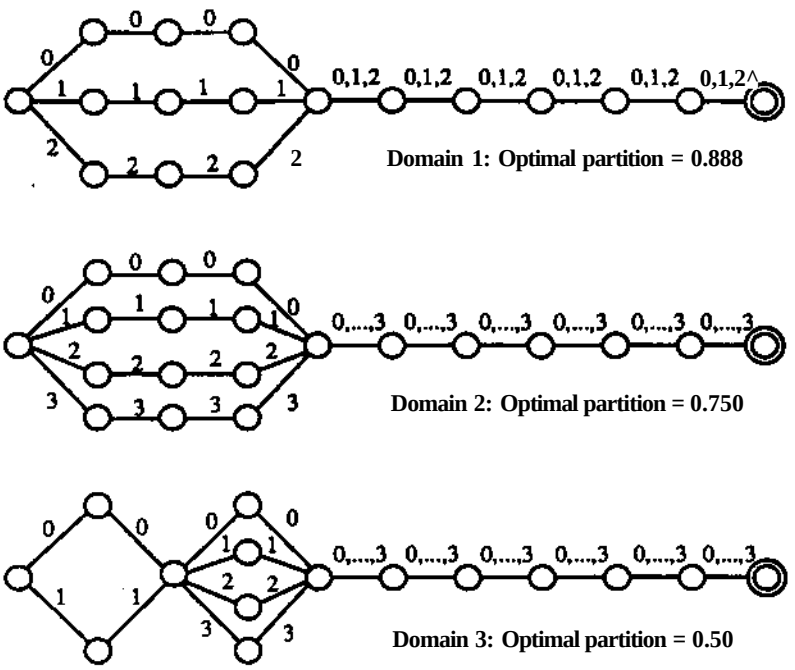

Figure 2: Artificial domains

performance on unseen examples.

The results of the experiments can be summarized as follows. The category utility *(CU)* of the top level classification was always higher than the *CU* of the opti nal partition of the domains given in Figure 2. In average, the following observation has been made: *CU distant* > *CU random* > *CU steepest*- The differences in the *CU* values were small.

The number of split and merge operator applications *(OA) ws* reduced by using either of the active selection techniques. In average, the following observation has been made: $0A_{Tan}dom \geq OA_{steepest} \geq$ *CU distant** The differences in the *OA* values were small.

Finally, the performance of the classification of examples was increased using either of the active selection techniques. The increased performance ranged from 0.1% in the simpler domain (Domain 1) to 3% in domain 2. The most-distant and the steepest-gradient selection resulted in similar results.

5.2 The domains of Pittsburgh bridges

The second experiment was conducted on the domain of Pittsburgh bridges. A detailed description of this domain appears in (Reich, 1991)[4]. We will only note that each of the 108 examples is described by a set of 7 specification properties and 5 design-description properties. Some of the examples are only

[4]**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 usemame and password. The files are in the /pub/machine-learning-databases directory.**

partially specified.

The testing **procedure** for this domain was as follows. A seed example initializes the process. A new example is **calculated** by the experimentation algorithm using *only the specification properties.* The **actual example learned** is selected as the best match from the data set; if more than one match exists, one is selected randomly). This example is discarded from future considerations. The experiment terminates when the data set is exhausted.

If the experiment would generate a learning curve where some of the examples are used for training and the remaining for testing the following behavior is expected. Initially, learning performance should improve compared to performance without experimentation. Later, performance should improve but gradually deteriorate since matching between the proposed example and the available examples will be worse as more examples are used and discarded, and since the examples remained as test cases are not typical of those learned; otherwise, they would have been chosen as training examples. To avoid this behavior we used the *nxV-CV* test[5] which was modified such that the hierarchy was created using the selection strategy.

The results of these experiments confirm the observations made in the tests of the artificial domains. The results, however, were not statistically significant.

5.3 Discussion

The problem of alleviating order effects on incremental learning is hard. The experiments performed show that small improvements of performance are obtained by usi ag various experimentation techniques. The small increase in performance suggest that These techniques, however, do not provide the optimal solution to the experimentation problem.

In many of the runs on the artificial domains, ECOBWEB has *gz* lerated approximate concepts at the first level. These approximations can easily be corrected by the hierarchy-correction scheme implemented in ECOBWEB (Reich and Fenves, 1991; Reich, 1991). In this process, examples that violate certain criteria (for example, their description contradicts characteristic values of parent nodes) are erased from the tree and re-learned. In most cases, this will place the examples in a more appropriate classes than their current location.

The behavior of ECOBWEB in these experiments can possibly be an artifact of the specific domains tested. For example, the artificial domains are too amenable to the random selection scheme of examples. In addition, the experimentation scheme tries to maximize the information extracted from the examples. In the beginning of the process, even irrelevant properties have some information that causes them to participate in the calculation of *ClTs* gradient. The use of large databases and more complex artificial domains can give better estimate to the ability of the approach.

The experiments and their results also point to the shallow understanding of the relations between

[5] **Scc (Reich, 1991), Appendix D for a detailed description of this and other statistical test procedures for evaluating learning systems.**

ECOBWEB's (as well as COBWEB'S) mechanisms and its subsequent behavior. There has not been any extensive study that tried to uncover the contribution of each of the system parts to the final system's performance.

6 Extensions and additional testings

One immediate extension is handling continuous properties in experimentation. This can be done by using our approach and calculating the gradient of *CU* by using two values: *current-value* $\pm$ *d*, where *d* is a pre-specified increment. Alternatively *d* can be dynamically calculated, for example from the standard deviation of the values of the continuous property.

Other extensions involve the understanding of the influence of different evaluations of the clustering on the ability to exploit experimentation. We hypothesize that a good evaluation maximizes such ability. An interesting evaluation method is the information measure of the classification (Boulton and Wallace, 1973) which can be used as an objective function to be maximized. In this view, additional examples lead to a more concise description of the domain. Since we are ultimately interested in the performance of the system after learning, a measure that correlates with performance will yield the best results.

Finally, using domain knowledge in the experimentation, as described in the previous section, and also as described by Kulkarni and Simon, 1988, can provide a much richer path for further extensions. In particular, knowledge can take into account the cost of experiments; it can analyze the benefits of experiments and suggest them dynamically; and it can integrate several experimentation scheme.[6]

7 Summary

We first elaborated on the problem of order effect in incremental leaming systems and provided several possible solutions. We have described a general experimentation scheme for concept formation tasks. Although the task is more complex than concept learning, the method is very simple. It only requires a metric for evaluating the quality of the knowledge structure before selecting the next example. The method was tested in several experiments that produced partially satisfactory results. Further extensions were suggested to improve the capabilities of method proposed.

We did not prove that the approach described is a *good* one; however the experiments conducted point to some important phenomena in the behavior of ECOBWEB (as well as of COBWEB) as a concept formation system and provides a path to explore experimentation methods for this task.

References

Amsterdam, J. (1988). Extending the Valiant model. In Laird, J., editor, *Proceedings of the Fifth International Conference on Machine Learning, Ann Arbor, MI*, pages 381-394, San Mateo, CA. Morgan Kaufmann.

[6] **The use of domain knowledge is a weak part in COBWEB, the role of such knowledge in obtaining enhanced design behavior is described in (Reich, 1991).**

Arciszewski,T., editor (1991). *Knowledge Acquisition in Civil Engineering.* American Society of Civil •Engineers, **New** York, (in press).

Boulton, D. M **and** Wallace, C. S. (1973). An information measure for hierarchical classification. *The* ***Computer Journal,*** **16(3):254-261.**

Buchanan, B. G., Sulivan, J., Cheng, T.-R, and Clearwater, S. H. (1988). Simulation-assisted inductive learning. In *Proceedings ofAAAI-88,* pages 552-557, St. Paul, Minnesota. Morgan Kaufmann.

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

Gross, K. P. (1988). Incremental multiple concept learning using experiments. In Laird, J., editor, ***Proceedings of the Fifth International Conference on Machine Learning,*** **pages 65-72, Ann Arbor,** MI. Morgan Kaufmann.

Kulkarni, D. and Simon, H. A. (1988). The processes of scientific discovery: the strategy of experimentation. ***Cognitive Science,*** **12(2): 139-175.**

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

Mitchell, T. M., Utgoff, P. E., and Banerji, R. (1983). Learning by experimentation: acquiring and refining problem-solving heuristics. InMichalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, ***Machine Learning: An Artificial Intelligence Approach,*** **pages 163-190. Tioga Press, Palo Alto, CA.**

Reich, Y. (1990). Design knowledge acquisition: Task analysis and a partial implementation. *Knowledge* ***Acquisition,*** **in Press.**

Reich, Y. (1991). ***Building and Improving Design Systems: A Machine Learning Approach.*** **PhD thesis,** Department of Civil Engineering, Carnegie Mellon University, Pittsburgh, PA.

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 UnsupervisedLearning,* pages 323-353, Los Altos, CA. Morgan Kaufmann.

Ruff, R. A. and Dietterich, T. G. (1988). What good*are experiments? In Laird, J., editor, *Proceedings* ***of the Fifth International Conference on Machine Learning,*** **pages 109-112, Ann Arbor, MI. Morgan** Kaufmann.

Russell, S. J. and Grosof, B. N. (1987). A declarative approach to bias in concept learning. In *Proceedings ofAAAI-87,* pages 505-510, Seattle, WA. Morgan Kaufmann.

VanLehn, K. (1987). Learning one subprocedure per lesson. *Artificial Intelligence,* 31(l):l-40.

Wisniewski, E., Winston, H., Smith, R., and Kleyn, M. (1987). A conceptual clustering program for **rule generation.** ***International Journal of Man-Machine Studies,*** **27(3):295-313.**