Artificial Intelligence for Engineering Design, Analysis and Manufacturing (1998), *12,* 95–106. Printed in the USA.
Copyright © 1998 Cambridge University Press 0890-0604/98 $12.50

Knowledge base for finite-element mesh design learned by inductive logic programming

BOJAN DOLŠAK,[1] IVAN BRATKO,[2] AND ANTON JEZERNIK[1]
[1]Faculty of Mechanical Engineering, University of Maribor, Slovenia
[2]Faculty of Computer and Information Science, University of Ljubljana, Slovenia

(Received June 27, 1997; Revised October 17, 1997; Accepted November 10, 1997)

Abstract

This paper addresses an important application of machine learning (ML) in design. One of the major bottlenecks in the process of engineering analysis by using the finite-element method—a design of the finite-element mesh—was a subject of improvement. Defining an appropriate geometric mesh model that ensures low approximation errors and avoids unnecessary computational overhead is a very difficult and time-consuming task based mainly on the user's experience. A knowledge base for finite-element mesh design has been constructed using the ML techniques. Ten mesh models have been used as a source of training examples. The mesh dataset was probably the first real-world relational dataset and became one of the most widely used training set for experimenting with inductive logic programming (ILP) systems. After several experiments with different ML systems in the last few years, the ILP system CLAUDIEN was chosen to construct the rules for determining the appropriate mesh resolution values. The ILP has been found to be an effective approach to the problem of mesh design. An evaluation of the resulting knowledge base shows that the mesh design patterns are captured well by the induced rules and represent a solid basis for practical application. The aim of this paper is not only to present the real-life ML application to design, but also to describe and discuss a relation of the work being done to the topic of this special issue: the proposed "dimensions" of ML in design.

Keywords: Finite-element Mesh Design; Inductive Logic Programming; Knowledge Base; Machine Learning

1. INTRODUCTION

The finite-element method (FEM), in the last 30 years the most successful numerical method in design, has been used extensively by engineers and modelling scientists to analyze stresses in physical structures. The effects of a specific loading case and support of such a structure can be expressed as a set of differential equations. However, it is not possible to solve such equations in a reasonable amount of computer time for arbitrarily complex structures. Therefore, we have to approximate the structure (Figure 1a) with a mesh model, that is, with a set of finite elements (FE) interconnected in the nodal points (Figure 1b).

Displacements in the nodal points then are adopted as the basic unknown parameters of the problem. A set of functions is chosen to approximate the displacement within each FE in terms of its nodal displacements. As a consequence of such a discretization, a set of linear algebraic equations, instead of differential equations, has to be solved. FEM is described in detail in Zienkiewicz and Taylor (1988).

An FE mesh should correspond to the geometric shape of the structure. In the areas where high deformations are expected, a fine mesh is required to ensure low approximation errors. On the other hand, a coarse mesh is adequate for the rest of the structure to avoid unnecessary computational overheads, since each additional FE adds new equations to the set that has to be solved. Thus, the "optimal" mesh model is the coarsest one that still ensures a satisfactory accuracy of the results. Figure 1b shows an example of that kind of mesh model (Dolšak, 1996).

Much experience and knowledge about FEM are required to know in advance where the mesh should be fine and where it should be coarse. A number of parameters, such as the shape of the structure, loads, and supports, have to be considered.

Reprint requests to: Dr. Bojan Dolšak, Faculty of Mechanical Engineering, Smetanova 17, 2000 Maribor, Slovenia. Tel: (+386)62-22-07-691; Fax: (+386)62-22-07-990; E-mail: dolsak@uni-mb.si

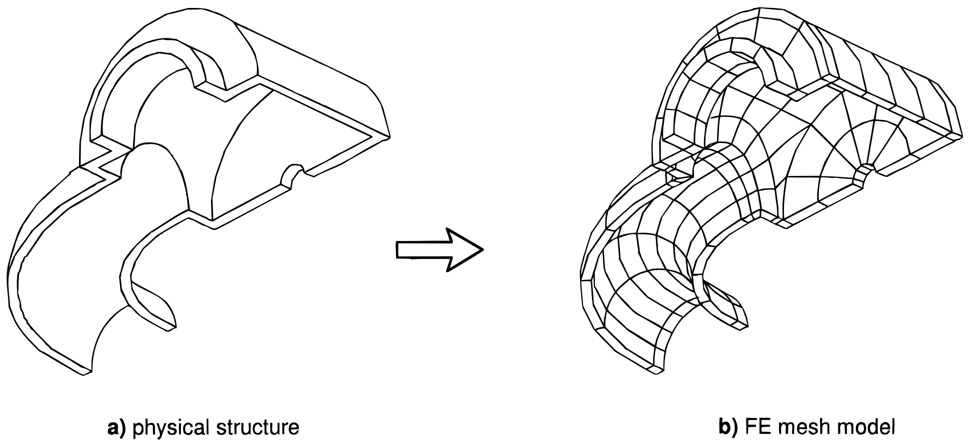

Fig. 1. FEM discretization. (a) Physical structure. (b) FE mesh model.

Most of the FEM packages on the market have the ability to build the FE mesh automatically. Yet, this option only considers the geometric shape of the structure, which is one of the influential parameters, but certainly not the only one and, in many cases, also not the most important one. It usually is necessary to design a few different meshes until the right one is found through experimentation. The trouble is that each mesh must be analyzed, since the next mesh should be generated with respect to the results derived from the previous one. If we consider that one FEM analysis can take from a few minutes to several hours and even days of computer time, there is obviously a great motivation to build an expert system (ES) that would be able to design "optimal" FE meshes in the first iteration, or at least in fewer trials.

The main question is, how to build a mesh design knowledge base (KB)? FEM has been applied extensively for the last 30 years. Still, there is no clear and satisfactory formalization of the mesh design know-how. Mesh design is still a mixture of art and experience. However, many published reports exist in terms of the problem definition, an adequate FE mesh (chosen after several trials), and the results of the analysis. These reports can be used as a source of training examples for machine learning (ML).

A typical procedure of applying ML tools has been carried out. This includes experimenting with several ML tools to explore the domain and the data, improving the dataset according to the shortcomings exposed by the initial experiments, detection of desirable domain-specific properties of ML tools, and, accordingly, the choice of a tool satisfying these properties for further improved application.

In this paper we concentrate on the last phase of this paradigm of ML application, describing the use of the inductive logic programming (ILP) system CLAUDIEN (De Raedt & Bruynooghe, 1993) that was chosen to construct rules for FE mesh design by generalizing the given examples. We directly address two "dimensions" of ML in design stated as a specific topic of this special issue. The paper clearly identifies one example of dimension 3 (what might be learned: rules for finite-element mesh design) and a corresponding example of dimension 5 (methods for learning: induction). Let us first, in the next section, present the learning problem and the structure of the training set.

2. LEARNING PROBLEM, EXAMPLES, AND BACKGROUND KNOWLEDGE

2.1. Source of examples

Reports about ten FEM analyses performed successfully in the past were used as a source for the training examples. Ten different FE mesh models, each used as the reference mesh model in one of these analyses, are comprised in the present training set. It is impossible to take into consideration all of the different structures that can appear in practice. Thus, the FE mesh models described in our training set represent a family with the following common features:

- all of the structures are cylindrical,
- loads are only due to forces or pressures (there are no thermal influences, etc.),
- highly local mesh refinement is not required.

A detailed description of the FE mesh models used can be found in Dolšak (1996). Figure 1b shows one of the models.

The FE meshes for all of the structures were "hand-constructed" and modified several times until the numerical results were accurate enough at acceptable computer time consumption.

Basically, a structure consists of a network of edges. A single edge can be described much more easily than the whole structure. Individual edges represent a lower level of the problem. The same type of edge can be found in many different structures, which allows us to describe a wide set of possible structures with a relatively small number of edge types. Therefore all of the FE mesh models were put together as a collection of edges. The edges were labeled by a combination of a letter and a number. Each letter (a–i) denotes a mesh model, while the consecutive numbers denote individual edges in the model. Figure 2 shows some labeled edges in the FE mesh model from Figure 1b.

The geometric form, loads, and supports are defined for all of the edges in the training set. In addition, certain important topological relations between the edges also are defined. To take into account this relational information among the edges, relational learning techniques apply most naturally to the learning about mesh design. Therefore, an ILP system was employed to build a KB for FE mesh design, since it realizes a form of relational learning.

According to the proposed taxonomy of the fourth dimension of ML in design (availability of knowledge for learning), the reports about FEM analyses that have been used as a source of the training examples are repositories of designs. The topological relations between the edges/examples might be considered as the interactions in the training set. Furthermore, another interaction takes place in the experiment described here. Since the topological relations between the edges of the structure are important in FE mesh design, relational learning was needed. One might analyze this as an example of a dimension 4 issue arising from interactions of dimensions 3 (what might be learned) and 5 (methods of learning). The interaction between these three dimensions seems to be the most important relation to design computing in general.

In general, the ILP problem statement involves positive and negative examples and background knowledge. In this section we develop a suitable representation for learning about FE mesh design in terms of examples and background knowledge.

2.2. Positive examples

The FE mesh is defined by stating the number of FEs along the edges. The relation to be learned is `mesh(E,N)`, where

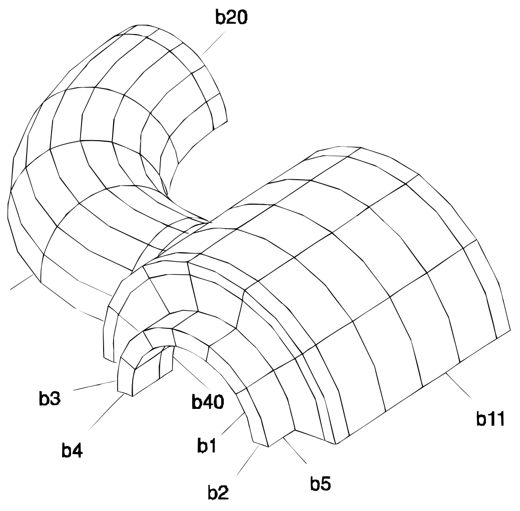

Fig. 2. Some labeled edges in the FE mesh model (Figure 1b).

`E` is the name of an edge in the structure and `N` is the recommended number of FEs along this edge. For each edge in the training set the number of FEs is given as a positive example. From the analysis point of view, the FE mesh models used to construct the training set are satisfactory. Yet, positive examples derived from these FE mesh models cannot be considered as perfect. To facilitate the learning process, some deviations were therefore also allowed. Thus, for `N` $\geq$ `8`, for each example `mesh(E,N)`, additional examples `mesh(E,N1), N1 = N1` $\pm$ `1` are added. We sometimes refer to the number `N` of FEs along an edge as the *class of the edge.*

Since edges with more than 12 FEs are quite rare, we decided to induce the rules only for classes from 1 to 12. Thus it may happen that the induced rules will not be able to classify all of the edges of the structure. Yet this should not be a serious problem, since one does not have to specify the number of FEs for all of the edges in order to build an FE mesh. In such cases, the FEM pre-processor constructs an FE mesh with respect to the given resolution values, using some built-in rules to determine the missing values.

2.3. Negative examples

Basically, negative examples were constructed according to the closed-world assumption. Therefore, the negative examples are constructed as the combinations of the names of the edges with all of the numbers between 1 and 12 other than those found in the positive examples. Again, some deviations from this are considered. For all of the edges with 5 to 7 FEs, deviations by one element are not stated as negative examples. For the edges with 8 or more elements, deviations by up to two elements are tolerated.

2.4. Background knowledge

Background knowledge contains definitions of the predicates that can be used in the hypotheses about the target relation `mesh/2.` It can be divided into two parts:

- attribute-value description of the edges,
- topological relations between the edges.

2.4.1. Attribute-value description of the edges

The following attributes of the edges that pertain to the FE mesh are described as background facts:

- the type of the edge using predicates: `long, usual, short, circuit, half_circuit, quarter_circuit, short_for_hole, long_for_hole, circuit_hole, half_circuit_hole, quarter_circuit_hole, not_important,`

- the supports using predicates:
  `free, one_side_fixed, two_side_fixed, fixed,`
- the loads using predicates:
  `not_loaded, one_side_loaded, two_side-_loaded, cont_loaded.`

There are special predicates that describe the type of an edge for those edges that have a circular shape (we have described a family of cylindrical structures). The other edges are classified by their length with the qualitative values "long," "usual," and "short." Special predicates also are used for the edges that are a part of a hole in the structure, since those places are usually important for the analysis. Short edges that do not belong to an important part of the structure are described with the predicate `not_important.`

To prevent unwanted displacements the structure must be supported. There are four predicates to describe the supports. Only the property of an edge being fixed (completely or at the endpoints) or free is considered, without distinguishing the supports with respect to the number of degrees of freedom.

Another influential parameter is the loading case. In fact, the purpose of the analysis is to inspect whether the structure is sufficiently dimensioned to sustain the loads during its exploitation. Similarly as for the supports, a description of a load only states the position of a load.

Each predicate that defines an attribute of an edge has one argument—the name of the edge. All of the edges in the training set are described by one predicate for each attribute. It could happen that an edge would be loaded with a combination of pressure (continuously) and force. In that case, two predicates can be used to describe the loading case of the edge. None of the training FE mesh models has an edge with a combined loading case; thus such an example does not exist in our training set.

2.4.2. Topological relations between the edges

A topological representation is needed because of the relationships between the edges that also pertain to the FE mesh. According to our experience there is also a connection between the edges that are neighbors or opposite (parallel) to each other, which affects the appropriate number of FEs on those edges. As we will see later, these two relations are indeed used very frequently in the induced rules.

A third interesting relation concerns the edges that are not only opposite but also have the same length or form. For instance, concentric circles have the same form. Such pairs of edges have been described using the predicate `equal.` All three predicates are binary.

An edge also can be a neighbor or opposite or equal to more than one edge. Because of that, it is best to use an ILP learning algorithm that allows nondeterminate literals. A literal such as `neighbor (a3,X)` is called nondeterminate because the "input" argument `a3` does not necessarily determine the "output" argument `X` (if `a3` has several neighbors, `X` can be any one of them). In spite of this, a "determinate" version of background knowledge description also was used in our earlier experiments (Dolšak & Muggleton, 1992; Dolšak et al., 1994) with the ILP program GOLEM (Muggleton & Feng, 1990), which is restricted to determinate literals only. CLAUDIEN allows nondeterminate literals.

2.5. Summary of the training set

The training set used in the present experiments comprises 4029 facts in the nondeterminate version of background knowledge and 644 positive examples. Distribution of the training examples considering the proposed predicates and target classes is believed to be sufficient.

The only proposed predicate that is not represented in the background knowledge is `quarter_circuit_hole.` Such edges are scarce enough, so this omission was considered noncritical.

3. SELECTION OF THE LEARNING ALGORITHM

The mesh dataset was probably the first real-world relational dataset. This contributed to the fact that mesh design became one of the most widely used domains for experimenting with ILP systems. Several learning systems have been applied to the FE mesh design problem in the last few years, most of them relational learning algorithms. As already mentioned, the first experiments with GOLEM were done using a smaller training set (Dolšak & Muggleton, 1992). After that, the training set was expanded and changed into the form that is described in Dolšak et al. (1994). Basically, the form of the training set was the same as discussed in the previous section, yet only the first five training models (a–e) were described in it. Following is a list of the ML experiments that were performed with such a training set:

- experiments with the ILP algorithm GOLEM (Muggleton & Feng, 1990), described in Džeroski (1991), Lavrač and Džeroski (1994), and Dolšak et al. (1994);
- experiments with the ILP algorithm FOIL (Van Laer et al., 1994), described in Džeroski (1991);
- experiments with the ILP algorithm mFOIL, described in Džeroski (1991);
- experiments with the ILP algorithm CLAUDIEN (De Raedt & Bruynooghe, 1993), described in Van Laer et al. (1994);
- experiments with the ILP algorithm MILP, described in Kovačič (1994);
- experiments with the ILP algorithm FOSSIL (Fürnkranz, 1994a), described in Fürnkranz (1994b);
- experiments with several attribute-value algorithms, described in Kononenko et al. (1994).

The results of these preliminary ML experiments to the FE mesh design problem were below expectations. The classification accuracy, measured by performing a leave-one-out test, was under 50% (12 to 44%). We believe that the reason for this lies in the nature of the training set; namely, each of the first five training models is in a way unique and has contributed a significant part of the examples in the training set. Because of this dissimilarity between the five structures, the leave-one-structure-out test was not very appropriate in this case.

To ensure the necessary conditions for the leave-one-out test, which is indeed the most natural test to validate the quality of the induced rules, a distribution of training examples was improved by the extension of the training set. Five additional structures (f–i) are included in the present training set. To ensure good results with ML with the new training set, the following features were considered to be important in the choice of an appropriate learning algorithm:

- relational learning,
- capability for dealing with nondeterminate literals,
- noise tolerance (both the number of positive and negative examples covered by the clause have to be considered),
- induction of practically useless rules should be prevented during the learning process.

The ILP system CLAUDIEN (De Raedt & Bruynooghe, 1993) has all of the desired properties. Furthermore, a set of negative examples (in our case 5157 instances) does not need to be presented to the system. Therefore, CLAUDIEN was chosen to be applied on the described new training set. A very useful feature of CLAUDIEN is that it allows the user to loosely specify the general form and contents of rules to be potentially generated by the induction algorithm.

4. LEARNING WITH CLAUDIEN

The learning of the rules for determining the resolution values for the FE mesh was carried out in six stages. In each learning stage a different form of rules was specified to be induced by CLAUDIEN. While specifying the form of the rules, the following criteria were considered:

- the use of all predicates from the training set has to be enabled,
- attribute description is needed for the edges introduced by geometric relations,
- the induction of too complicated rules should be avoided—the rules should not contain more than two geometric relations.

The whole training set was used in each learning step; thus, the order of the learning steps is not important and has no influence on the induced rules whatsoever.

The learning process was further guided by setting the required accuracy (percentage of positive examples among those covered) and coverage (the number of covered ground "cases"; this is understood to be the number of substitutions from the training set that make the body of the induced clause true; therefore, a case does not necessarily correspond to an example).

In the following subsections all six learning stages are described. We give examples of induced rules and rule schemes. All have a Prolog syntax as generated or accepted by CLAUDIEN.

We also refer several times to the CPU time spent to perform a particular learning step. The whole learning process was carried out using the computer hardware SUN Sparc Station where CLAUDIEN was installed. The computer used can be considered as a representative powerful workstation. Thus there exists not only a strong correlation between CPU time and computational complexity of the learning process, but also a solid basis for comparison with some other ML applications.

4.1. Learning step 1: Class rules without relations

In the first stage CLAUDIEN induced 17 classification rules that contain only attribute descriptions of the edges. The following rule schema was specified for this purpose:

```
clausemodel('mesh(Edge1,
[1,2,3,4,5,6,7,8,9,10,11,12])<-\
  +1{(Type(Edgel),Support(Edgel),
     Load(Edgel)}').
```

This says that the conclusion part of an induced rule is a literal of the form mesh/2, specifying up to 12 FE on Edge1. The condition part of the rule may mention at least one and at most three attributes of Edge1 (Type, Support, and Load). For example, rule number 10 with 100% accuracy (`perc_cov(1)`), which was induced in little more than 14 CPU seconds, is

```
rule(10,[perc_cov(1),body(4),
cpu(14.1667)],
  (mesh(Edgel,11) :-
    long(Edge1),
    one_side_loaded(Edgel))).
```

It specifies 11 FEs on the long edges, loaded at one side. From the rule description it also can be seen that there are four substitutions in the training set that make the body of the rule true.

4.2. Learning step 2: Class rules with one relation

In the second stage of the learning process the use of a single topological relation was allowed. To prevent the induction of practically useless rules, the following conditions also were specified by the language for the target hypothesis:

- the rules must contain at least one attribute description of the actual edge;

- the new edge, introduced by the topological relation, has to be further described by at least one attribute:

```
clausemodel('mesh(Edgel,
[1,2,3,4,5,6,7,8,9,10,11,12]) <-\
  <+1{Type(Edgel),Support(Edge1),
    Load(Edge1)},\
  {<Relation(Edgel,Edge2),\
  +1{Type(Edge2),Support(Edge2),
    Load(Edge2)}>}').
```

The predicate variable Relation was defined as one of: neighbor, opposite, or equal. Limited by this specification, 351 rules were induced in the second learning stage. Here is an example of a rule that specifies 7 FEs for the "usual" edges, which have a continuously loaded and both sided fixed neighbor of the quarter circuit geometric form:

```
rule(340,[perc_cov(1),body(4),
cpu(19375.2)],
  (mesh(Edgel,7) :-
    usual(Edge1),
    neighbor(Edgel,Edge2),
    quarter_circuit(Edge2),
    two_side_fixed(Edge2),
    cont_loaded(Edge2))).
```

4.3. Learning step 3: Class rules with two relations, both referring to the target edge

Classification rules with two topological relations were induced in two learning stages. First, rules with two topological relations, both referring to the "target" edge, were built. For example, the following rule, which was induced in the third learning stage, contains two topological relations (neighbor and opposite) with an actual edge (Edge1) as the first argument in both cases:

```
rule(166,[perc_cov(0.909091),body(22),
cpu(1026)],
  (mesh(Edgel,9) :-
    half_circuit(Edge1),
    not_loaded(Edge1),
    neighbor(Edge1,Edge2),
    usual(Edge2),
    opposite(Edge1,Edge3),
    half_circuit(Edge3),
    not_loaded(Edge3))).
```

CLAUDIEN induced 1988 rules of this type in 128207 CPU seconds.

4.4. Learning step 4: Class rules with two relations in chain

The search space was even more complex in the fourth stage of the learning process, when the rules with two topological relations, describing a chain of edges, were sought. Despite the fact that the required accuracy and coverage of the rules were increased (Table 1), almost 300,000 CPU seconds were spent in the fourth learning step to build 1700 rules. An example is

```
rule(1535,[perc_cov(1),body(8),
cpu(64226)],
  (mesh(Edge1,3) :-
    cont_loaded(Edge1),
    neighbor(Edge1,Edge2),
    half_circuit(Edge2),
    cont_loaded(Edge2),
    neighbor(Edge2,Edge3),
    not_important(Edge3),
    one_side_loaded(Edge3))).
```

In this case, two new edges, Edge2 and Edge3, are introduced as a chain of neighbors together with the target edge, Edge1.

4.5. Learning step 5: Interval rules with one relation

In the fifth learning stage we allowed the induced rules to specify more than one class. The use of one topological relation was also allowed. In this case we considered all 17

Table 1. *Learning the rules—basic parameters*

Step	Form of the rules	Accuracy	Coverage	Rules	CPU sec.
1	Class rules without relations	≥0.90	≥3	17	79
2	Class rules with one relation	≥0.90	≥3	351	26657
3	Class rules with two relations (Edge1)	≥0.90	≥3	1988	128207
4	Class rules with two relations in chain	≥0.95	≥10	1700	299833
5	Interval rules with one relation	≥0.98	≥20	395	894573
6	Limiting rules considering the edge type	1	≥8	11	8666
				4462	1356237

classes that appear in the training set. On the other hand, the required accuracy and coverage of the rules were increased further. CLAUDIEN spent more than 10 days of CPU time to induce 395 rules that meet the given specification in this stage of learning. Most of the rules specify an interval over the classes, although there are also examples with some missing classes, for example:

```
rule(362,[perc_cov(1),body(26),
cpu(775726)],
  (one(A);two(A);four(A) :-
  mesh(Edge1,A),
    one_side_fixed(Edge1),
    neighbor(Edge1,Edge2),
    free(Edge2),
    cont_loaded(Edge2))).
```

The rule specifies 1, 2, or 4 FEs for edges that are fixed on one side and have a neighbor that is free and continuously loaded.

4.6. Learning step 6: Limiting rules considering the edge type

In the last stage, interval specifications were again allowed; however, this time only the type of the edge was considered in the body of a clause. For each type description predicate, one rule was induced to specify the possible numbers of FEs corresponding to that type of edge. Thus, for example, the following rule says that short edges in the training set have between 1 and 4 FEs:

```
rule(2,[perc_cov(1),body(109),
cpu(13.9333)],
  (one(A);two(A);three(A);four(A) :-
  mesh(Edge1,A),
    short(Edge1))).
```

4.7. Summary of the learning process

The basic parameters of the overall learning process are presented in Table 1. CLAUDIEN induced 4462 rules altogether in some 15 days of effective CPU time on a SUN Sparc Station. More than one-half of this time (66%) was spent for induction of only 395 rules (9%) in the fifth learning step, when interval rules with one geometric relation were learned! A computational complexity of the rest of the learning process was quite reasonable, considering the number of induced rules. The learning process could be terminated after a specified CPU time; however, this option was not applied, since there is no connection between the order of the induced rules and their quality. The induced rules mention all of the background predicates, as well as all the classes represented in the training set.

5. POSTPROCESSING OF THE INDUCED RULES

The induced rules were found to be inappropriate to be placed directly into KB. Many of them were subsequently eliminated, while the form and order of the remaining rules were adjusted to meet the application requirements. In postprocessing, the rules that specify an exact number of FEs were treated separately from the interval rules.

Rules were eliminated considering the following conditions:

- if they covered less than three positive examples,
- if they were duplicated,
- if they were subsumed by a more general rule for the same class,
- if they covered only additional positive examples,
- if they were merged with other rules that had the same body but specified different classes (in the case of interval rules).

The first condition seems to be redundant, since the minimum number 3 of substitutions from the training set that make the body of the clause true has already been considered in the learning process. Yet it must be taken into account that the number of substitutions is not always the same as the number of positive examples covered by the rule. One positive example may cause more than one successful substitution. For example, one edge can have up to four neighbors with the same attributes.

After the learning process, 2686 rules (60%) were eliminated in total. The remaining 1776 rules were simplified. To ensure the best possible efficiency, all of the elements of the rules that are not necessary for practical application (for example, CPU time spent for induction quoted in the head of the rules) were abandoned. On the other hand, a mechanism for preventing the infinite loops was implemented, to handle the recursive rule that was induced in one of the previous learning experiments (Dolšak et al., 1994) and was added to the KB.

In the FEM pre-processor environment one has to define the exact number of FEs on each edge of the structure to be idealized. Therefore, rules for determining the most appropriate class from the proposed list of classes, specified by the interval rules, were also added to the KB. Basically, these rules compare the geometric type of two edges and give the difference between the average classes used in the training set for that kind of edge. For example, the following rule says that usual edges have on average 6 FEs less than long edges:

```
compare_type(Edge1,Edge2,-6) :-
  usual(Edge1),
    long(Edge2).
```

The comparison rules were built on the basis of simple statistics on the training set. The comparison rules in the

KB enable the inference engine to choose the class out of the proposed list of classes by considering the type of the opposite edge. If there is no opposite edge, a single class is calculated as an arithmetic mean value of the proposed classes.

All of the rules in the KB were ordered to optimize the effectiveness and accuracy of the ES. Because of the top-down search strategy implemented in Prolog, which was used as a programming language for the ES shell, the most reliable rules were placed on top of the KB. This is especially important in the case of classification rules, to ensure that the best rule is executed to determine the number of FEs for each particular edge of the structure. Several criteria were used to define a proper order of the classification rules in the KB:

- Rules that classify single classes were placed in the KB before interval rules to minimize the need for selection of the most appropriate class from the proposed list of classes with the comparative rules. For the same reason, the recursive rule is placed just after the rules for single classes.
- Rules without negative covers had priority.
- Rules describing the type of the actual edge were placed before the rest of the rules, since the type of the edge (relative length) is most obviously related to the number of FEs.
- Rules with 100% accuracy (on training data) were ordered from simple attribute rules to more complex rules with one and two geometric relations. The opposite ordering was used for the rules that cover negative examples.

6. RESULTS

6.1. Knowledge base and expert system shell

Finally, 1873 rules and 31 facts were accepted into the KB as described in the previous section. It is the most comprehensive KB of all of the previous experiments in this field that were mentioned before. Undoubtedly the main reason for this is the extension of the training set. Yet, the extended training set was not the only difference between the experiment described here and other ML experiments carried out in this field in the past. The following features are characteristic of our recent experiment and were performed for the first time:

- Induction of practically useless rules was prevented by prescribing the form of the rules.
- Rules that specify more than one single class also were induced.
- The learning process was not stopped until all of the rules that satisfied the given criteria were induced.
- The ILP system CLAUDIEN was adjusted to deal more efficiently with such an extensive training set.

It also should be noted that some additional rules were put into the KB after the learning process.

The KB is written in Prolog syntax in the form of Prolog rules and facts. Thus it is sufficiently transparent and can be easily extended if necessary. Most of the rules (1711 = 91%) contain geometric relations. Furthermore, the classification rules without geometric relations cover only 110 positive examples (17%). Thus, it is quite obvious that the geometric relations are even more important than was expected.

The ES shell also is written in Prolog. It facilitates the proper use of the KB for FE mesh design, as well as communication between the user and the system. One of the more interesting features of the user interface is its capability to explain the inference process.

6.2. Evaluation of the expert system

A comprehensive evaluation of the ES was carried out. First, the classification accuracy was tested in various ways. Furthermore, an additional, more informative criterion of success, called classification cost, was introduced. This takes into account the size of error in the case of misclassification. Misclassification cost was defined as:

$$\text{Cost} = |N1 - N2|/\max(N1, N2),$$

where $N1$ is the reference number of FE and $N2$ is the number prescribed by the ES. Misclassification cost is normalized between 0 and 1, where cost means better classification performance. In our domain, the worst possible error is to classify a 1-element edge into "class" 17; this would incur a misclassification cost of $|17-1|/17 = 0.94$.

The results of the ES have been used in practice as the resolution values for real-life mesh generation with the FEM pre-processor. The results of these tests also are presented in this section.

6.2.1. Test on the edges from the training set

First, the ES was used to determine the number of FEs for all of the edges from the training set. The classification accuracy was 78.26%, and the misclassification cost was 0.092.

6.2.2. Leave-one-out test

Here the classification accuracy and cost were measured for each training model, which, for that purpose, was effectively eliminated from the training set. Because of its time complexity, the learning process was not actually repeated each time with a different training set. Instead, the rules that could not be induced if the current structure would not be a part of the training set were eliminated from the KB. A detailed description of the elimination process can be found in Dolšak (1996). Although this process does not guarantee exactly the same results as would be obtained by repeated induction without the eliminated structure, any differences in results are unlikely. The classification accuracy was between 40.48 and 80.46%, and 59.09% on average. The lowest misclassification cost for a single structure was 0.064, and the highest was 0.244.

The results of the ES for the "eliminated" structure also were used as basic parameters for mesh generation with the well-known FEM pre-processor within the popular computer-aided design package I-DEAS Master Series™ (I-DEAS, 1993). Let us consider the structure with the lowest classification performance as an example. To meet the requirements of the built-in mapped meshing method, the structure was partitioned into six subvolumes (Figure 3).

In spite of quite low classification accuracy, the mesh built on the basis of the results of the ES (Figure 4a) is not bad, at least as an initial attempt. A comparison with the reference, a manually designed mesh (Figure 4b), which was used for the numerical analysis, shows that the ES proposed a few more elements on some of the edges, but the overall pattern of the mesh was almost the same.

6.2.3. *Tenfold cross-validation test on randomly selected subsets*

The training set was randomly divided into ten subsets. The elimination of each subset from the training set was simulated in the same way as in the case of the leave-one-out test. On average, 70.16% of the edges were classified correctly, and the average misclassification cost was 0.127.

6.2.4. *Test on an unseen structure*

In the final test, the ES was employed to specify the mesh resolution values for a completely new cylindrical structure that was not included in the training set. The performed classification accuracy was 86.67%, with a misclassification cost of 0.028, which is quite impressive considering the results of the previous ML experiments on this problem. There were only four misclassified edges (encircled in Figure 5a). For all quarter circuit edges the ES proposed one FE more than in the reference design, yet this qualified as an allowable deviation according to the construction of our training set.

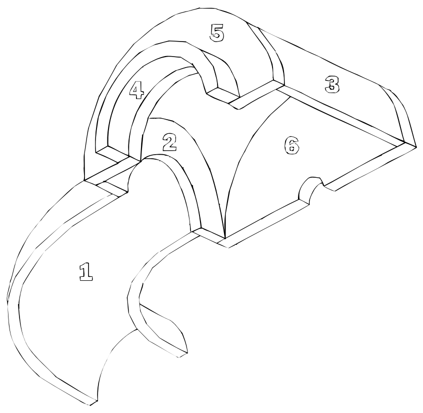

Fig. 3. Partitioning of the part for mapped meshing procedure.

With respect to the results of the ES, the FEM pre-processor built almost the same FE mesh (Figure 5b) as that designed manually (Figure 5c). As a matter of fact, the knowledge-based designed mesh model has a few FEs more. We can suspect that it also would require a little more CPU time for analysis, but this is not the most important issue here. The fact is that the ES proposed an almost "optimal" mesh model in the first trial. On the other hand, the mesh model that was produced by automatic mesh design (free meshing), which is implemented in the I-DEAS Master Series™ FEM pre-processor (Figure 5d), cannot be considered as a near-"optimal" trial. Moreover, in this particular case, the FE mesh model is not good enough even to represent properly the geometric shape of the structure.

Again, a partitioning of the part into smaller subvolumes with a maximum of six surfaces was done to satisfy the requirements of the mapped meshing procedure. This test also shows that classification accuracy may be misleading as a measure of success in this domain. The reason is that the mesh generator will automatically correct some errors. This happened in the case of Figure 5 as follows. The mapped meshing technique assumes an equal number of FEs on the opposite sides of the surfaces. The FEM pre-processor put 14 FEs on the edges while the ES specified 11 FEs, since the total number of FEs on the opposite side is also 14.

7. CONCLUSIONS

We have presented an ML application to the real-world engineering design problem. The extensive KB for FE mesh design was built by using the ILP system CLAUDIEN. Yet, the KB is not only the result of our experiment. It is also a demonstration of the strength of ILP on this particular task as well as its potential in general. According to the practical experience with FEM systems, it would be very difficult to extract the knowledge about FE mesh design from the human expert. Much experience is required to design an "optimal" FE mesh, which is hard to describe explicitly. On the other hand, the inductive knowledge acquisition turned out to be an effective approach to solve the FE mesh design problem. Furthermore, human experts also were asked to look over the induced rules and to assess their meaningfulness in comparison with their own expert knowledge. The most important overall conclusion of the expert evaluation of the induced rules was found to be that the form of the rules is as expected and in general matches the knowledge used by human experts.

Considering the results of the overall KB evaluation, it can be concluded that the KB represents a solid basis for practical application. The mesh generated according to the ES results for the example "eliminated" training model is insignificantly different from the reference mesh. Although the results of the ES seemed rather low in that particular case, the comparison of performance measured by the leave-one-out test and by the tenfold cross-validation test indicates a satisfactory distribution of the training examples. The

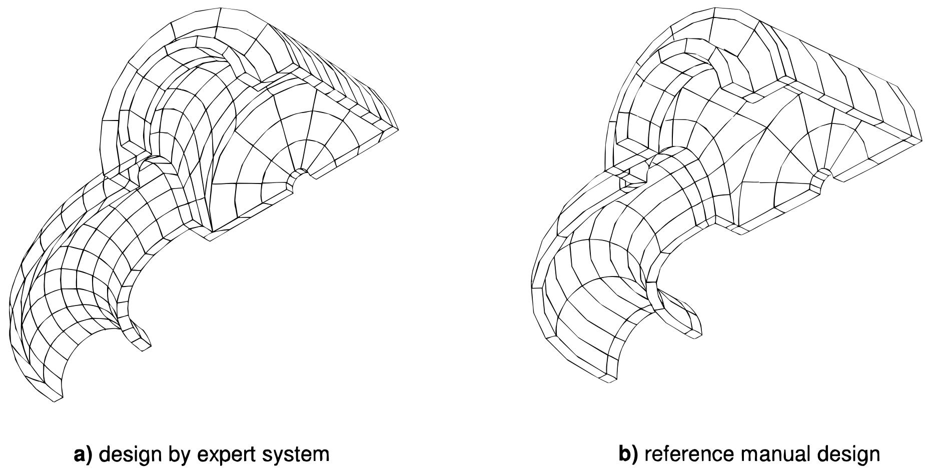

Fig. 4. A comparison between knowledge-based and manual mesh design. (a) Design by expert system. (b) Reference manual design.

results of the cross-validation test are somewhat higher, yet the difference is relatively small. The applicability of the ES was further confirmed by the test on the completely new structure.

To simplify the learning problem, the training set was designed with the aim of being representative of a particular type of structures. However, the KB presented here also can be applied as a general tool for determining the mesh resolution values for structures outside the scope of these types. These values have to be adjusted subsequently according to the specific requirements of the particular analysis and suited to the method for the mesh generation.

The ES application enabled the determination of an almost "optimal" FE mesh for a structure within the type scope in the first attempt. Such effectiveness of the ES cannot be expected in cases of structures of different types. Yet, the resolution values specified by the ES can always serve as a basis for an initial FE mesh, which is subject to further adaptations by considering the results of the numerical analyses. It is very important to choose a good initial mesh and minimize the number of iterative steps, leading to an appropriate mesh model. The ES presented can be very helpful in performing this task, especially to inexperienced users. Finally, by minimizing the number of mesh models over multiple FEM analyses, even the extensive learning time also can be amortized.

When the rules in the KB specify the list of the proposed classes, "the most appropriate" class should be determined. In spite of the special attention that was paid to the methods for determining "the most appropriate" class out of the proposed list, the interval rules in the KB still enable better results than those that have been achieved. It is a matter of

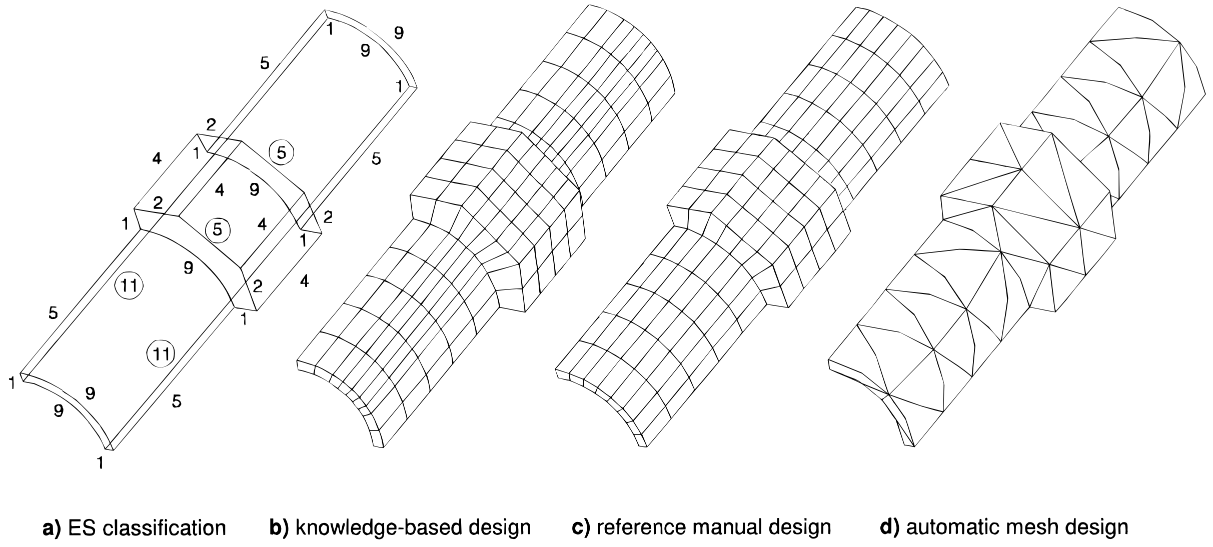

Fig. 5. Test on the new structure. (a) ES classification. (b) Knowledge-based design. (c) Reference manual design. (d) Automatic mesh design.

further research to find more appropriate rules for determining a single class out of the proposed list of classes.

Finally, let us relate the work described in this paper to the proposed "dimensions" of ML in design. We already pointed out the important relation between dimensions 3, 4, and 5. In any case, the method of learning (dimension 5) is dependent on the form of the available knowledge (dimension 4), which is strongly related to the domain concept that is going to be learned (dimension 3).

With respect to the first dimension (what can trigger learning?), we can give two reasons for learning the rules for FE mesh design: first, a need and motivation to improve the ability of the existent FEM software, and, second, the nature of domain knowledge based mostly on experiences that cannot be expressed explicitly enough to build the KB. According to the evaluation results, it can be concluded that the consequence of learning (dimension 7) is the improvement of the design process.

We believe that there is another important "dimension" of ML in design which has not been set up in the call for this special issue. Are the present learning methods and algorithms adequate enough to be applied to design? In our experiment, ILP was found to be a powerful and useful tool for learning the FE mesh design rules. And yet, there still exists substantial room for further improvements. Despite some changes and adaptations that have been made to CLAUDIEN to deal with the FE mesh design problem, the extensive and time-consuming postprocessing of the induced rules still is necessary.

ACKNOWLEDGMENTS

This research was financially supported by the Ministry of Science and Technology, Republic of Slovenia, and by the European Community within the action for co-operation in science and technology with Central and Eastern European countries (PECO92)—ILPNET, contract no. CIPA3510OCT920044. Researchers from several ILPNET nodes have been involved in ML applications to the FE mesh design problem and made their learning programs available for that purpose. S. Muggleton's GOLEM and L. De Raedt's CLAUDIEN were particularly important in our work. We would like to acknowledge Professor T.K. Hellen of Imperial College of Science, Technology and Medicine, London; M. Prašnički and R. Kogler from TAM Research & Development Institute, Maribor; and S. Ulaga from the Faculty of Mechanical Sciences, Maribor, for providing the training examples. Special thanks to Professor Hellen for his willingness to carry out the expert evaluation of the induced rules.

REFERENCES

De Raedt, L., & Bruynooghe, M. (1993). A theory of clausal discovery. *Proc. Thirteenth Int. Joint Conf. on Artificial Intelligence*, pp. 1058–1063. Morgan Kaufmann, San Mateo, CA.

Dolšak, B. (1996). *A contribution to intelligent mesh design for FEM analyses* (in Slovene, with English abstract). Ph.D. Thesis. Faculty of Mechanical Engineering, University of Maribor, Slovenia.

Dolšak, B., Jezernik, A., & Bratko, I. (1994). A knowledge base for finite element mesh design. *Artificial Intelligence in Engineering 9*, pp. 19–27.

Dolšak, B., & Muggleton, S. (1992). The application of inductive logic programming to finite element mesh design. In *Inductive Logic Programming* (Muggleton, S., Ed.), pp. 453–472. Academic Press, New York.

Džeroski, S. (1991). *Handling noise in inductive logic programming.* M.Sc. Thesis. Faculty of Electrical Engineering and Computer Science, University of Ljubljana, Slovenia.

Fürnkranz, J. (1994a). FOSSIL: A robust relational learner. *Proc. European Conf. on Machine Learning*, Catania, Italy. Springer-Verlag, New York.

Fürnkranz, J. (1994b). Top-down pruning in relational learning. *Proc. 11th European Conf. on Artificial Intelligence,* Amsterdam. The Netherlands, pp. 453–457.

I-DEAS Master Series^{TM} (1993). *Exploring I-DEAS Simulation*, 2nd ed. Structural Dynamic Research Corporation. Milford, OH, USA.

Kononenko, I., Šimec, E., & Robnik, M. (1994). *Overcoming the myopia of inductive learning algorithms.* Report No. FER-LUI-1/94. Faculty of Electrical Eng. and Computer Science, University of Ljubljana, Slovenia.

Kovačič, M. (1994). MILP—A stochastic approach to inductive logic programming. *Proc. Fourth Int. Workshop on Inductive Logic Programming (ILP-94)* (Wrobel, S., Ed.), pp. 123–138. GMD-Studien Nr. 237, Germany.

Lavrač, N., & Džeroski, S. (1994). *Inductive Logic Programming: Techniques and Applications.* Ellis Horwood, New York.

Muggleton, S., & Feng, C. (1990). Efficient induction of logic programs. Technical Report TIRM-90-044, The Turing Institute, Glasgow, Scotland.

Van Laer, W., De Raedt, L., & Dehaspe, L. (1994). *Applications of a logical discovery engine.* Report No. CW 195. Department of Computing Science, KU Leuven, Belgium.

Zienkiewicz, O.C., & Taylor, R.L. (1988). *The Finite Element Method—Basic Formulation and Linear Problems.* McGraw-Hill, London.

Bojan Dolšak is a senior lecturer of computing and CAD related subjects at the Faculty of Mechanical Engineering, University of Maribor, Slovenia. He received his B.Sc., M.Sc., and Ph.D. from Faculty of Mechanical Engineering at the University of Maribor. He has a deep and thorough understanding of the modern programming techniques and computer graphics. He is also concerned with the artificial intelligence methods, especially expert systems and machine learning. His current interests include computer aided design in general, three-dimensional modeling, engineering analyses using FEM, artificial intelligence methods in general, expert systems for CAD and machine learning.

Ivan Bratko is professor of computer science at the Faculty of Computer and Information Sc., Ljubljana University, Slovenia. He heads the AI laboratories at J. Stefan Institute and the University. He is the chairman of ISSEK, International School for the Synthesis of Expert Knowledge. He has conducted research in machine learning, knowledge-based systems, qualitative modelling, intelligent robotics, heuristic programming and computer chess. His main interests in ML have been in learning from noisy data, combining learning and qualitative reasoning, and various applications of ML and ILP. Ivan Bratko's numerous publications include the books PROLOG Programming for Artificial Intelligence (Addison-Wesley) and KARDIO: a Study in Deep and Qualitative Knowledge for Expert Systems (MIT Press).

Anton Jezernik is a professor of computing and CAD related subjects and a head of the Laboratory for Technical Software at the Faculty of Mechanical Engineering, University of Maribor, Slovenia. He received his M.Sc. and Ph.D. from Imperial College of Science, Technology and Medicine, University of London. After nearly three years stay at electricity generating industry in the UK, he has been working since 1973 at the University of Maribor, where he carried out many research projects mainly from the field of computer aided design. His current main interests are feature-based design and expert systems for CAD.