Learning rules for finite element mesh design

Finite element methods are used extensively by engineers and modelling scientists to analyse stresses in physical structures. These structures are represented quantitatively as finite collections of elements. The deformation of each element is computed using linear algebraic equations. In order to design a numerical model of a physical structure it is necessary to decide the appropriate resolution for modelling each component part. Considerable expertise is required in choosing these resolution values. Too fine a mesh leads to unnecessary computational overheads when executing the model. Too coarse a mesh produces intolerable approximation errors.

The resolution of a FE mesh is determined by the number of elements on each of its edges. The problem of learning rules for determining the resolution of a FE mesh is therefore, to learn rules that determine the number of elements on an edge. The data here is from experiments conducted with *Golem* [**?**] (reported in [**?**]). The task is to learn rules for the number of elements using the following information:

- The training examples have the form **mesh(Edge,Number_of_elements)**, where **Edge** is an edge label (unique for each edge) and **Number_of_elements** is the number of elements on the edge denoted by label **Edge**. The number of elements on an edge varies from 1 to 17.
- The background knowledge describes some of the factors that influence the resolution of a FE mesh, such as the type of edges, boundary conditions and loadings, as well as the shape of the structure (relations of neighborhood etc.).
- According to its importance and geometric shape, an edge can belong to one of the following types: **important_long**, **important**, **important_short**, **not_important**, **circuit**, **half_circuit**, **quarter_circuit**, **short_for_hole**, **long_for_hole**, **circuit_hole**, **half_circuit_hole** and **quarter_circuit_hole**.
- With respect to the boundary conditions an edge can be **free**, **one_side_fixed**, **two_side_fixed** or **fixed**.
- According to the loadings an edge is **not_loaded**, **one_side_loaded**, **two_side_loaded** or **continuously_loaded**.
- Background knowledge about the shape of a structure includes the symmetric relations **neighbour/2** and **opposite/2**, as well as the relation **equal/2**.

The data concerns five structures labelled "a" – "e". For each structure, data files are as used in the original *Golem* experiments. That is, background knowledge files have a ".b" suffix, positive example files have a ".f" suffix, and negative example files have a ".n" suffix. The files are all stored in one compressed TAR file.

References

[1] Dolsak B. and Muggleton S. (1992). The application of Inductive Logic Programming to finite element mesh design. In S. Muggleton editor. *Inductive Logic Programming*, Academic Press, London.

[2] Muggleton S.H. and Feng C. (1990). Efficient induction of logic programs. *Proceedings of the First Conference on Algorithmic Learning Theory*, Tokyo.