OBJECT-ORIENTED PROGRAMMING FOR TOPOLOGY DESIGN OF TRUSSES BY A GENETIC ALGORITHM

K. THEERAKITTAYAKORN[1], P. NANAKORN[1*†], P. PAKLACKSAME[1], T. MEKSANG[1] and C. WANGSITTIKUL[1]

[1]*School of Civil Engineering and Technology, Sirindhorn International Institute of Technology, Thammasat University, Thailand*

ABSTRACT

This paper presents object-oriented programming for topology design of trusses by a genetic algorithm (GA). When a GA is used to design a truss structure, intermediate solutions during the evolution process are always evaluated against the employed design code by the finite element method. This design process by a GA requires a computer program that can perform finite element analysis and also has the necessary information of the employed design code and standard. Efficient implementation of this kind of program requires excellent data structuring. In this study, the object-oriented programming concept is used to design program structures for topology design of trusses by a GA. The proposed structures use classes to represent design codes and standards. The structures also include classes of truss design elements that are created by combining the ordinary truss finite elements with the necessary design information via class inheritance. The truss-element objects of these derived classes are therefore capable of knowing whether they satisfy the design constraints or not. This significantly improves the implementation of GAs for design of trusses.

Keywords: Topology design, truss, object-oriented programming, genetic algorithm.

1. INTRODUCTION

Topology design of truss structures by genetic algorithms (GAs) has been investigated by many researchers (Deb and Gulati 2001; Rajan 1995). The ultimate goal of these research works is to create GA topology design methods that can be used in real design applications with real design codes and standards. If GA design methods are to be used in real design, it is essential that efficient programming technologies are used in order to allow easy incorporation of design codes and standards. In fact, efficient programming technologies will also facilitate the research and development of GA design methods.

The object-oriented programming (OOP) concept considers programs as the communication between entities called objects. Each of these objects is in fact a collection of encapsulated data and

* Corresponding author: Email: nanakorn@siit.tu.ac.th

† Presenter: Email: nanakorn@siit.tu.ac.th

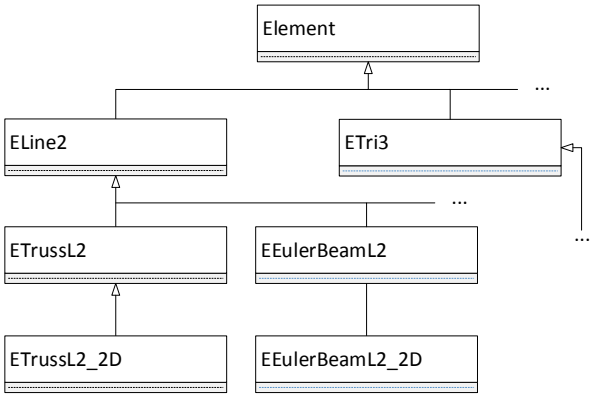

Figure 1: A class diagram for finite elements.

their behavior. The OOP technique allows data to be structured efficiently without alteration of their physical meanings. For example, matrices and their operations in a program written with the OOP technique look like matrices and their operations in the standard mathematical language. Programming using the OOP concept is natural and easy. Maintaining and extending this kind of program also become simple since the programs are easy to understand and their parts can also be reused efficiently (Weisfeld 2008).

The OOP technique has been used in programming of some GA applications for truss optimization (Krishnamoorthy et al. 2002; Sivakumar et al. 2004). In this study, OOP is employed in such a way that design codes and standards can be easily incorporated in topology design of trusses. In order to achieve that, classes in C++ are created to represent the entities related to structural analysis, design codes, and standards. Multiple-inheritance is then used to create new classes of design elements that are finite elements with embedded knowledge of design codes and standards. Each design element can evaluate itself whether it passes design requirements of the employed design code when design loads are given to the element. In addition, it also has knowledge of steel sections and materials that are available in the employed standard. When a GA is used in topology design of trusses, each individual in the GA is in fact a truss, which has to be evaluated against the employed design code in order to determine its fitness. The proposed design elements allow this evaluation to be performed efficiently. In this paper, the models of classes for these design elements are presented and discussed.

2. CLASSES FOR STRUCTURAL ANALYSIS AND DESIGN INFORMATION

In order to develop the classes for design elements, their base classes have to be developed first. These base classes are those classes for structural analysis and design information and they can be grouped into three main groups, namely classes for finite elements, classes for design codes, and classes for design standards. The classes for finite elements are for analysis of structures while the classes for design codes and standards are for design of structures. These three classes will be used together to create derived design element classes.

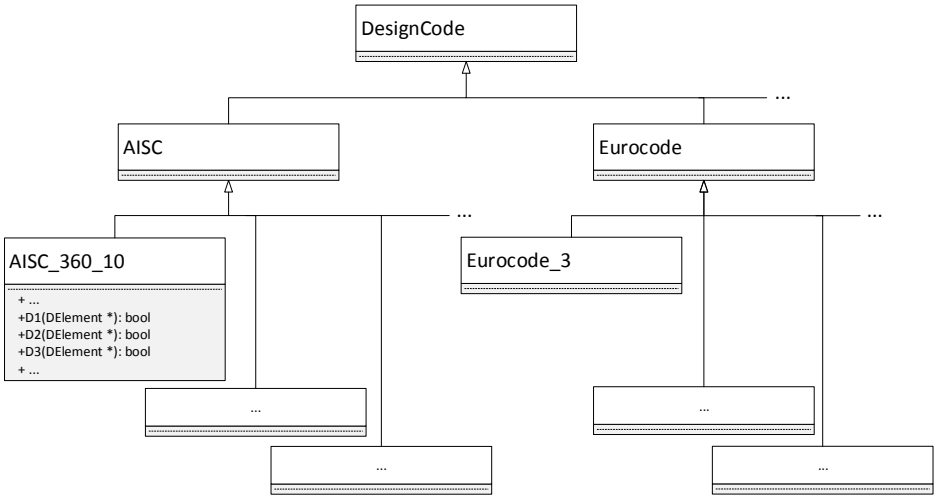

Figure 2: A class diagram for design codes.

2.1. Classes for finite elements

Any GA program for topology design of trusses always requires the finite element method (FEM). In any finite element program that employs the OOP concept, several classes will be used to represent various entities in FEM. For example, there must be classes that represent vectors, matrices, nodes, elements, materials, and geometry. The existing element classes in the employed finite element program can then be used directly without any modification in the development of a GA program for topology design of trusses. The complete hierarchy of element classes can be quite involved because of a large number of element types. Therefore, only an example class diagram showing only a few types of element, including truss elements, is shown here in Figure 1. The *Element* class in the diagram serves as an abstract base class of all element types. The *Element* class is used to maintain the information that is common among all types of element. It is usually convenient to derive, from the *Element* class, a layer of abstract classes based on different element geometry and numbers of nodes. For example, the *ELine2* and *ETri3* classes in Figure 1 will be used as base classes for line elements with two nodes and triangular elements with 3 nodes, respectively. In the figure, the *ETrussL2* class that is the class for the 2-noded truss element is derived from the *ELine2* class and the *ETrussL2_2D* class for the 2D 2-noded truss element is derived from the *ETrussL2* class. These *ETrussL2* and *ETrussL2_2D* classes will be used to create the design elements for topology design of trusses.

2.2. Classes for design codes

Figure 2 shows an example class diagram for design codes. The hierarchy in the figure allows different codes to be grouped based on the real world arrangement. In the figure, the *AISC_360_10* class represents the ANSI/AISC 360−10 specification while the *Eurocode_3* class obviously represents the Eurocode 3. The basic concept of creating classes for design codes to be used in a genetic algorithm is to use these classes to check whether designs are satisfactory or not with respect to all the relevant design requirements. Assume that the ANSI/AISC 360−10 specification is

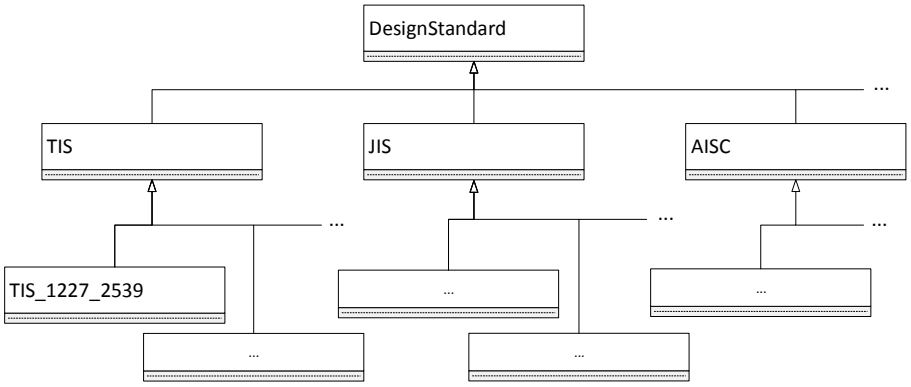

Figure 3: Class diagram for design standards.

used to design a truss structure. An object of the *AISC_360_10* class will be created but the object will not be used to directly determine the optimal design of the truss. Instead, the *AISC_360_10* object will actually work like a design checker who examines whether each design is satisfactory or not. The GA will utilize the information from the *AISC_360_10* object to find the optimal design of the problem. The design checking is performed by public member functions of the *AISC_360_10* class. Figure 2 shows that the *AISC_360_10* class has public member functions and some of them, i.e., D1, D2, and D3, are explicitly shown by names. These public member functions take a pointer of a design element as its argument and return a Boolean result. In fact, each of these functions corresponds to a section in the ANSI/AISC 360−10 specification. For example, the D2 function corresponds to the D2 section in the specification, which is concerned with tensile yielding and tensile rupture of tension members. To use the D2 function, the design element under consideration is sent to the D2 function via its pointer argument. The function then uses the information in the design element to check whether the element satisfies all requirements in the D2 section of the specification or not. If it does, the function returns true; otherwise it returns false. In case that the requirements are irrelevant to the member sent, the function can just return true. For example, if a compression member is sent to the D2 function, the function will return true. Public member functions can be created for, if practical, all sections of each specification.

2.3. Classes for design standards

Design standards in this study mean standards for design sections and materials. Figure 3 illustrates an example class diagram for design standards. Similar to design codes, standards can be grouped based on the real existing groups. As an example, Figure 3 shows the *TIS_1227_2539* class that represents the Thai Industry Standard for hot rolled structural steel sections. Figure 4 shows the details of the *TIS_1227_2539* class. In the figure, two classes called the *DSteelSection* and *DSteelMaterial* classes are shown. The *DSteelSection* class is used to maintain the geometrical information that is common among all types of section while the *DSteelMaterial* class is used to maintain the information related to standard steel material properties. These two classes, including their derived classes, will be available for all design standards. Figure 4 also shows some derived

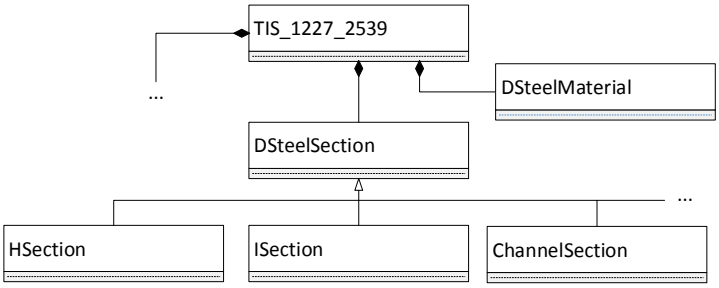

Figure 4: Class diagram of a Thai Industrial Standard.

classes of the *DSteelSection* class. They include the *HSection*, *ISection*, and *ChannelSecion* classes that represent the geometrical details of H, I and Channel sections, respectively. Every design standard class will need to have member objects of these derived classes for sectional geometry in order to list all sections available in the standard being represented. In summary, each design standard class provides the available sections and materials for a GA to use in topology optimization of trusses.

3. CLASSES FOR DESIGN ELEMENTS

The classes for truss finite elements, design codes, and design standards, when combined together, allow the structural analysis capability and design information to be utilized efficiently together. This is done by simply deriving new classes of elements, called design elements, from these classes. Figure 5 shows an example class diagram of design elements for truss problems. The *DElement* class serves as the base class for the design element classes. Objects of the *DElement* class use objects of the design code and design standard classes to gain access to the design information. The *DTrussL2* class is derived from the *DElement* and *ETrussL2* classes. As a result, an object of the *DTrussL2* class is a truss finite element that has the information of the design code and design standard used. Similarly, the *DTrussL2_2D* class is derived from the *DElement* and *ETrussL2_2D* classes. Its object is therefore a 2D truss finite element with the knowledge of the design code and standard.

When a GA is used to perform topology design of a truss, it starts its search from many design solutions in the search space at the same time. The algorithm needs to evaluate every design solution in its population in order to create a new generation of better design solutions. In topology design of structures, all structures in a GA population must be evaluated and compared. In the evaluation of structures, FEM is always used to analyze the structures. After that, the obtained finite element analysis results will be evaluated against the employed design code. Without classes specifically designed for these tasks, the GA will need to obtain the analysis results from a finite element module and bring those results to another module that evaluates the results against the employed design code. With the design elements discussed above, the whole process can be efficiently performed without separate modules. In addition, since each design element also has direct access to the employed design standard, the design element can easily extract any geometrical

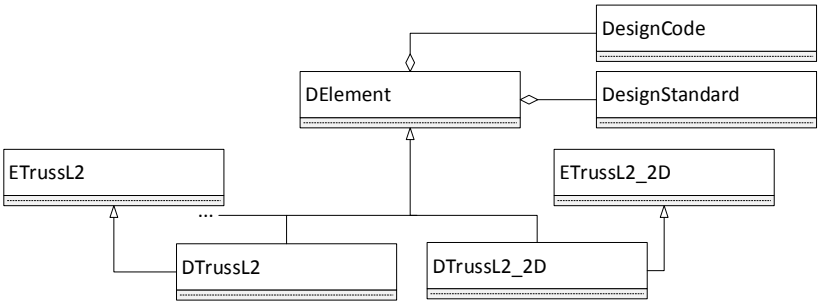

Figure 5: Class diagram of design elements.

and material properties from the design standard when needed. This greatly facilitates the GA process both during the evaluation of current design solutions as well as the creation of new design solutions.

4. CONCLUSIONS

In this paper, object-oriented modelling for topology design of trusses by a GA is shown. In the topology design of a truss by a GA, a design code and a design standard are used. All design solutions in each GA generation need to be evaluated against the employed design code. In addition, all design solutions are created by employing sections and materials given in the employed design standard. In this study, classes for design codes and standards are developed. They are subsequently used with classes for truss finite elements to develop truss design elements, each of which represents a truss finite element with the information of the selected design code and standard. This kind of element allows GAs for topology design of trusses to be implemented efficiently.

5. ACKNOWLEDGMENTS

Kasem Theerakittayakorn is supported by the Royal Golden Jubilee PhD Program, the Thailand Research Fund.

REFERENCES

Deb K, and Gulati S (2001). Design of truss-structures for minimum weight using genetic algorithms. Finite Elements in Analysis and Design. 37(5). pp. 447-465.

Krishnamoorthy CS, Prasanna Venkatesh P, and Sudarshan R (2002). Object-oriented framework for genetic algorithms with application to space truss optimization. Journal of Computing in Civil Engineering. 16(1). pp. 66-75.

Rajan S (1995). Sizing, shape, and topology design optimization of trusses using genetic algorithm. Journal of Structural Engineering. 121(10). pp. 1480-1487.

Sivakumar P, Rajaraman A, Knight GMS, and Ramachandramurthy DS (2004). Object-oriented optimization approach using genetic algorithms for lattice towers. Journal of Computing in Civil Engineering. 18(2). pp. 162-171.

Weisfeld M (2008). The object-oriented thought process. Addison-Wesley Professional. Third Edition.