INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI
RICE UNIVERSITY

Problem Generator System in
Engineering Design Tutor

by

Yuan Rui

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

APPROVED, THESIS COMMITTEE:

Michael Terk, Assistant Professor, Chair
Civil and Environmental Engineering

Ahmad J Durrani, Professor
Civil and Environmental Engineering

Willy Zwaanepoel, Hasselman Professor
Computer Science Department

HOUSTON, TEXAS
December, 2001
ABSTRACT

Problem Generator System in Engineering Design Tutor

by

Yuan Rui

The Internet has provided an ideal environment for engineering education. This thesis describes the development of a new problem generator used in the Engineering Design Tutor (EDT) system to provide multi-step problem generation capability. The EDT system allows the user to specify a set of design concepts from a specific domain, based on which a problem tree is generated. All the possible solution paths in the tree are searched and the most complete and correct path is automatically selected as the shortest path. Using the values obtained from a constraint satisfaction solution for the variables in the shortest path, and the templates provided for the problem description, the system allows the automatic generation of a set of problems all using the same shortest solution path.
Acknowledgements

First, I would like to thank my advisor Dr. Michael Terk for his enormous technical and moral support. I am confident that without his enthusiasm, this research work would not have been complete. I would like to thank Dr. Ahmad J. Durrani and Dr. Willy Zwaenepoel for serving on my Thesis Committee. I owe thanks to my fellow graduate students for their help and valuable discussions.

Last, but not least, I would like to express my gratitude to my parents and my brother for the support, encouragement and help that they have given me, even though being miles away.
# Table of Contents

1. Overview
   1.1 Introduction
   1.2 Motivation
   1.3 Organization

2. Introduction to Problem Generator System
   2.1 Description and Criteria
      2.1.1 Correctness
      2.1.2 Completeness
      2.1.3 User-Friendliness
      2.1.4 Interaction
   2.2 Existing Problem Generator Systems
      2.2.1 WebTest
      2.2.2 WinAsks
      2.2.3 QuestionMark
      2.2.4 CAPA
   2.3 Comparison of Existing Problem Generation Systems
   2.4 Conclusions

3. Introduction to Engineering Design Tutor System
   3.1 Engineering Design Problems
      3.1.1 Problem Representation
      3.1.2 Multi-disciplinary Elements
      3.1.3 Iterative Solutions
      3.1.4 Multiple Problem Types
   3.2 Methodology of EDT
   3.3 EDT Architecture
      3.3.1 The Domain Knowledge Component
      3.3.2 The Problem Generator Component
   3.4 Limitations of EDT

4. Problem Generator System in EDT
   4.1 Problem Generation in EDT
   4.2 SASE Knowledge Representation
      4.2.1 Data Item
      4.2.2 Decision Table
      4.2.3 Information Network
   4.3 EDT Knowledge Representation
      4.3.1 Problem Tree
      4.3.2 Solution Path
   4.4 Automated Problem Generation in EDT
      4.4.1 Problem Generation and Selection
      4.4.2 Current Implementation
      4.4.3 Solution Path Weight
      4.4.4 Templates
      4.4.5 Multi-Problem Generation
      4.4.6 User-Guided Generation of Problems
List of Tables

Table 2.1 Problem Types 16
Table 4.1 Regions of a Decision Table 29
Table 4.2 Example of Decision Table 30
Table 4.3 Example of a Problem Template 43
List of Figures

Fig 2.1 User-friendliness Illustration 7
Fig 3.1 Structure of EDT 23
Fig 4.1 Information Network 33
Fig 4.2 Problem Tree 35
Fig 4.3 Solution Path 36
Fig 4.4 Previous Implementation 38
Fig 4.5 Weight Method 41
Fig 4.6 Intermediate Point 45
Fig 4.7 Ratio Point 46
Fig 5.1 Problem Definition 48
Fig 5.2 Variables 50
Fig 5.3 Constraint Output 51
Fig 5.4 Templates 52
Fig 5.5 Sections 54
Fig 5.6 Solution Key 55
Fig 5.7 Problem Saving 55
Fig 6.1 Error Message 59
Chapter 1

Overview

1.1 Introduction

The practice of civil engineering is evolving rapidly. New technologies and new methods of performing civil engineering tasks are constantly being developed. The rapid advances occurring in computer software and hardware have provided the engineers with a powerful means of processing, storing, retrieving, and displaying data. This has made computer science a growing and essential part of nearly every field of engineering, including the practice of civil engineering. This rapid implementation of computer technology in civil engineering has presented a wide array of new research problems in the field.

The engineering education has taken advantage of the developments to continuously improve engineering education. Simulation software, expert systems, computer visualization are some of the example of computer technology that has been used to enhance engineering education. Recently, the increased availability of Internet and the development and adoption of the World Wide Net (WWW) technology has creates significant new resource for a variety of application and has resulted in increased interest in adapting it for engineering education.

The Internet and WWW have a number of characteristics that makes them suitable for supporting educational activities. For example, email allows students and educators to communicate much more often and at provides a mechanism for reaching the instructor
outside the pre-set office hours. Instructors can use the WWW to provide students with unlimited access of constantly updated lecture notes.

The Internet also serves as a forum to promote group discussion, which is time and distance independent. There are many forms, which this group discussion can take. These include video conferencing, where by the use of a small video camera and microphone members of the group can actually see and hear each other.

The Internet also serves as a useful advertising tool for different centers of education, as it allows them to provide information about courses, fees and other useful information to a global market. The advantage of this is that this information can be updated quite easily and at minimal cost and inconvenience. It also provides an instant medium for interested parties to communicate with these institutions.

The Internet can also be used as a very effective administrative tool to collect and store information. Here the Internet is used as the front-end application that allows the user to enter details, which can then be stored in a database. This information can then be accessed and manipulated by a select number of users for whom access rights and permissions have been granted. Most educational institutions have started to use this for a variety of tasks, including the storing of student records, academic results, financial information and other sensitive data. The advantage of this is the Internet provides a medium to collect and view the data from anywhere there is a computer linked to the Internet.

The advantages discussed above reveal the Internet to be the ideal environment for introducing new educational technologies to many disciplines.
1.2 Motivation

The main purpose of engineering education is to instill in students the basic engineering principles and to instruct them in applying them to real engineering problems. In civil engineering, students usually achieve this goal by drill-and-practice mode of learning, i.e., working on numerous similar problems. The instructor has to provide a number of similar exercises to allow the student the ability to evaluate the mastery of concepts demonstrated in these exercises.

In order to achieve this purpose, instructors must create a large database of problems for the students. As a result, the drill-and-practice approach to teaching is very costly in terms of instructor’s time and workload. Also, some measures must be taken to deal with the varying achievement levels of different students. So the drill-and-practice approach to teaching often fails to challenge or evaluate the learning process of many students.

Currently, the most common method to meet the drill-and-practice demands for materials is to develop a database of problems. Problems are created and stored in the database. The drawback of this method is, due to the limited quantity of problems in the database, after multiple uses by multiple people, the chance for the same problem to be chosen is quite large.

An educational use of Internet has made this problem even more acute, since university are not considering expanding through distance education, thus increasing the class size. As a result, a number of researchers have developed automated problem generation systems. These systems are aimed at simplifying the task of generating and grading problems for freshmen and sophomore level introductory classes. These classes
service a large number of students and have typically relied on multiple-choice exams to reduce the grading burden. The new generation of automated systems has been shown to be very successful in simplifying the problem generation, grading and tracking for multiple-choice problems.

This thesis deals with developing and extending this concept to support multi-step problems that are common in upper level engineering courses. The problem generator presented in this thesis can be used by instructors who wish to generate customized multi-step homework that tests the students on routine engineering design concepts. Students can also use it to support the self-paced evaluation of their knowledge of the subject area. This work extends and enhances the Engineering Design Tutor (EDT) system that was developed at Rice University [1].

1.3 Organization

This thesis is organized as follows: Chapter 2 provides a survey of existing educational software and generalizes the criteria required for an educational testing software; Chapter 3 introduces the Engineering Design Tutor (EDT) system which aims to simplify the task of generating and grading multi-step engineering problems; Chapter 4 introduces new algorithms and functionalities that were developed in this thesis to improve the problem generation facilities of EDT; Chapter 5 illustrates the operation of the new implementation of the problem generator module of EDT with a case from steel beam design; Chapter 6 provides an evaluation of the system performance; and Chapter 7 summarizes the contribution of this thesis and proposes future directions for this work.
Chapter 2

Introduction to Problem Generator System

2.1 Description and Criteria

We define a problem generator system as an interactive program that supports the creation and grading of problems. The generator system should provide users access to a large supply of practice problems and should also support the extension of the problem database by developers. The functionalities should include:

- problem generation
- solution generation
- import and export of previously generated solutions

There are four specific criteria to judge if a system achieves the above functions:

- Correctness
- Completeness
- User-friendliness
- Interaction

Each of these criteria is discussed below.

2.1.1 Correctness

Correctness refers to the match between the problem and the underlying domain that the problem is attempting to measure. Any problem generated by the system should be
related to the subject matter, clearly worded, and understandable. Problems are generated to test some particular skill or a specific body of knowledge by representing the contents and themes relevant to that skill. For example, a structural design problem must draw from many aspects of structural design and test the user's skill and knowledge of multiple relevant concepts. Problems that ask students about related, but tangential information interfere with this goal. Correctness is usually achieved by clear wording. If the wording is vague, the skills employed by the user may not correspond to those the problems were generated to test. They should test the learners' knowledge of the subject matter, not their ability of grammar and vocabulary.

2.1.2 Completeness

Completeness refers to the range of materials covered by the problem. A complete problem should be clearly defined, contain all the information necessary to achieve a solution, and have a consistent solution. Usually, a clear and detailed description of the subject matter (topics) and range of skills to be tested is provided before a problem is generated. In the problem generator system, users first define the design concepts, and the problem to be generated must include every specified concept. A problem would not be complete if it did not address all specified concepts.

For a set of design concepts, multiple problems could be generated. They might include only the design concepts defined, or might include some other relevant design concepts. Some technique must be available to select the most complete problem for any set of concepts. For example, the most complete design problem would be defined as the one that best matches the goal by only using the specified concepts.
2.1.3 User-Friendliness

User-friendliness is often a key factor in the success of the system. No matter how correct and complete the system may seem from the developer's perspective, it may still be of little use to the user if it cannot convey its knowledge or expertise effectively.

According to "Quality Criteria for Computer-delivered Tests" [2], user-friendliness implies the software should:

- be consistent in operation;
- be easy to use, especially for candidates who are not experienced computer users;
- have screens which are consistent in appearance and legible;
- give candidates easy access to any information which is required for a problem and which forms part of the software.

Fig. 2.1 User-friendliness Illustration
Here is an example to illustrate user-friendliness. In this example, a tree is used to display the hierarchy of some concepts in the civil engineering field. This kind of representation has been quite familiar to Windows users (it’s like the interface used in the windows File Manager). The idea – to use representation familiar to users – guides user’s attention to the content of the display, rather than the format of the interface. Also, it helps the user feel more comfortable with the software and easier to accept the information presented.

2.1.4 Interaction

Appropriate and immediate information - the interaction between users and systems - should be given to the users during the problem generation (i.e. after each step, in completion of the generation). An ideal form of interaction is to allow the user to observe the problem generation, inform users of the current situation, and obtain any information necessary to complete the problem generation successfully.

Immediate feedback from the system is important for users to finish the task efficiently. For example, most calculations during the problem generations are fast, but there can be some time-consuming calculations. If a user who is not familiar with the generation process sent out the command to start one of these calculations, but no feedback was provided to the user about the current status, he could assume that the command was not received or accepted. Then he sent out the commands more times to ensure the system could respond, which might cost more time or even cause the system to break down. From above, immediate feedback or indications about the command received is essential to the success of the system. A more informative feedback might
include information on which command the system has received, what it is doing and how much time is required.

2.2 Existing Problem Generator Systems

This section provides a brief description of the usage and status of current existing problem generator systems. They share some common features, although each has unique characteristics.

2.2.1 WebTest [3]

WebTest is a tool developed for delivery of assessments to large numbers of students via the Internet. It has many potential uses, particularly in all area of education and in training for business and industry.

A key feature of WebTest is randomization. Randomization of problem choice and of parameters in problems is used to generate different, but equivalent instances of problem sets. The following example shows two instances of the same problem.

<table>
<thead>
<tr>
<th>There is one parameter (the number) that is chosen at random from integers between –9 and 9, excluding 0. The answers will be derived from the randomized parameters.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Differentiate $x(x+3)$</td>
</tr>
</tbody>
</table>

This feature allows the same problem set to be re-used by a student as a tool for the generation of examples. It reduces the risks of plagiarism and cheating that would arise if students were all given the same test.
Automatic feedback and results can be given immediately, and used by the student as the basis for further study. Problems are stored in an on-line database and can be secured by restricting delivery to specified dates and particular computers on the Internet.

Another feature of WebTest is the inclusion of the specification Document Type Definition (DTD), which is independent of the WebTest engine. The DTD structures the problems in a logical way which is efficient for delivery through the WebTest engine, but still enables diversity in problem design, a high degree of randomization, the inclusion of mathematical display and a wide variety of problem styles and answer judging mechanisms. It also does not restrict the user to web delivery, since the problems are defined independently of the physical representation and can be output in several forms.

2.2.2 WinAsks [4]

WinAsks is a program that creates, manages and analyzes multimedia questionnaires and surveys. It has been studied particularly for an educational use and its features make it ideal for problem set creation.

The creation of problem sets through WinAsks required only that the users to enter the text of problems and answers, leaving the program to select the required options. Problems are customizable and users can change the style (e.g. fonts, colors, ..), or other options (e.g. time, errors, hints..). Problems could include images, sounds and animations. The check of each given answer can be made automatically by the program. Other available options include the use of created problem sets, the inclusion, insertion, linking of other files, and the use of comments. After the problem sets are created in the local machine, users can send the sets to an on-line collection center for all problems created.
by WinAsks. This allows users to share the problems they've created and thus all users draw from a larger selection of problem. Problems can be exported in HTML format for use on the Internet.

### 2.2.3 QuestionMark [5]

QuestionMark is an on-line web-based testing and tracking system. It helps users to create problems in a Windows 95/98/NT/2000 environment, and then to store them on a web server.

QuestionMark enables users to quickly create problem sets from problem banks organized by topic. Each problem set is generated by randomly selecting a specified number of problems from each topic. Problems may be answered randomly or in strict order according to the author's preference. The author can also set the program to select problem from the problem libraries in a random fashion, so that each time a test is administered, the test-taker essentially gets a different test. Furthermore, the order in which the randomized problems are presented can be shuffled for each test-taker. These features make the possibility of test-takers sharing information with others less of a concern.

The software automatically grades each problem as it is given and informs the user of the result. Immediate feedback is available to users, including the success or failure and how much time is spent on each problem.

A style library is available from which to choose an extensive number of templates and screen layouts for creating new problems. The user can set screen attributes such as text size, position of text boxes, and borders.
2.2.4. CAPA [6]

CAPA is an on-line system for learning, teaching, assessment and administration. It provides students with problem sets, quizzes, or exams.

First a prototype problem set is developed in a language that includes substitution of random numbers for variables and codes the solutions based on these variables. The level of differences among these personalized problems is entirely controlled by the user when the prototype problem set is written. It is relatively easy to have several variables with large ranges significantly reducing the possibility that two problems will have the same answer. The problem sets are also reusable. Each student’s answers are unique but the concepts and principles covered in a problem are the same for all students.

Students get immediate feedback and relevant guidance. Timely feedback is valuable in learning, and CAPA informs students promptly whether they have understood the concepts and were able to apply them correctly. In addition, summaries of success and failure on problems are available on-line to the instructor. Thus, he/she can review the concepts and methods of solution relevant to the difficulties encountered. The system also provides information on the progress and performance of a particular student so that early action can be taken to address problems.

2.3 Comparison of Existing Problem Generator Systems

One common feature of the systems above is the wide use of the Internet. An on-line library/database exists for all the systems as a collection of problems generated. The library/database can be organized by specific categories. Users can retrieve the problems
via the Internet and solve them on-line. This provides great convenience and avoids the waste of redundant work already done by others.

In the systems discussed, the generation procedure is quite straightforward. Problems are generated by replacing values for the variables in the problem template in the local machine, and then stored back to the database server. This procedure can be called a static one. In some systems, such as CAPA, the generation may involve large set of variables and the answer may be defined as an expression that uses the values of those variables. Moreover, the solution procedure varies from problem to problem and a student has to get used to applying the correct design concept depending on the problem. This procedure is referred to as dynamic problem generation. Specifications for the problem sets are stored in a specification web server. When a problem is to be generated, instructors retrieve the information essential to that problem from the specification web server and generate problems in the local machine. Then the problems are stored into the database server.

Another feature shared by the systems is the ability to generate a range of problem types. Design of a system includes the choice of appropriate problem types that will best fulfill the goal. The following is the definitions of some problem types listed in Table 2.1.

- Numeric problems: a participant is prompt to enter a numeric value.

  How many players are there in a baseball team?

- Word response: the participant types in a single word or a few words to indicate the answer.
The film "Pulp Fiction" was directed by

- Multiple response: similar to multiple choice except the participant is not limited to choosing one response; he/she can select none, one or more of the choices offered.

Select the actors who appeared in "The Untouchables".
- Harrison Ford
- Al Pacino
- Martin Sheen
- Sean Connery
- Andy Garcia
- Kevin Costner

- Essay problem: the participant answers by typing some amount of characters of text.

Describe the relationship between Hamlet and Laertes from Shakespeare's "Hamlet". Include examples of their differences and similarities in dispositions and modes of behavior.

- Matching/ranking (selection problem): a series of statements are presented and the participant can select from a number of items and answers.
Starting with the first to take office, put these U.S Presidents in order:
Lyndon B Johnson
Bill Clinton
Richard Nixon
Ronald Reagan
Franklin D Roosevelt
Abraham Lincoln

- Hotspot: a participant clicks on a picture to indicate their choice. Dependent upon their choice certain feedback will be assigned.

Place the marker at 12 o'clock.

- Matrix: this problem type presents several multiple-choice problems together where the participant selects one choice for each statement or problem presented.

Which actor starred as James Bond in "GoldenEye"?
- Pierce Brosnan
- Roger Moore
- Sean Connery
- Timothy Dalton

Which actor starred as James Bond more than any other actor?
- Pierce Brosnan
- Roger Moore
- Sean Connery
- Timothy Dalton

Which actor starred as James Bond in "License to Kill"?
- Pierce Brosnan
- Roger Moore
- Sean Connery
- Timothy Dalton

- Explanation screens: insert text or graphics for the participant to view prior to answering a series of problems.
Place the text markers inside the relevant boxes to identify the planets of our solar system. A point will be awarded for every correct answer.

<table>
<thead>
<tr>
<th></th>
<th>WebTest</th>
<th>WinAsks</th>
<th>QuestionMark</th>
<th>CAPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple Choice</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Numeric Problem</td>
<td>×</td>
<td></td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Word Response</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>True/False</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Matching/Ranking</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Essay Problem</td>
<td></td>
<td></td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Multiple Response</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fill-in-the-blank</td>
<td></td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Hot Spot</td>
<td></td>
<td></td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>Matrix</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Explanation Screen</td>
<td></td>
<td></td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

Table 2.1 Problem Types
These common features are very helpful to the problem generator system design. Besides, some other features are also helpful, such as the style template of QuestionMark and the specification DTD of WebTest.

2.4 Conclusions

The review of the existing problem generation systems shows the range of functionality that can be supported in such system. At this time none of these tools provide support for multi-step problems that form the core of the drill-and-practice approach common in upper level engineering problems. The next chapter describes the Engineering Design Tutor (EDT) system that attempts to provide similar problem generation capability but is targeted at multi-step design problems.
Chapter 3

Introduction to Engineering Design Tutor System

The various problem generator systems introduced before can’t support engineering design domains because they can only generate problems with single-step solutions. However, engineering design problems are usually multi-step. Thus, it’s necessary to develop a new automated problem generator in the engineering design domain. Engineering Design Tutor (EDT) is such a software package that incorporates teaching and engineering design. It’s the outcome of hard work of many people. Its first generation has established the general framework, which includes the basic functionalities required by an engineering problem generator. This chapter provides a general overview of the EDT system [1].

3.1 Engineering Design Problems

To better understand the difference between EDT and other problem generators, it is useful to examine the features of engineering design problems first.

3.1.1 Problem Representation

Engineering design requires the development of a system, which will satisfy a set of specifications in a determined discipline under some special conditions. “This implies that a major aspect of engineering design is the development of ways to describe these specifications clearly, concisely, and unambiguously. This tells us that representation is an important design issue, although perhaps with an emphasis on the translation of the original design objectives and constraints into some versions of the specification
language, and on the recognition that the specifications provide the starting point for a construction or fabrication process."[7] The set of specifications used must be robust enough to handle any set of special conditions, requirements or systems.

### 3.1.2 Multi-disciplinary Elements

An engineering discipline may decide a basic concept in its domain. Even the simplest engineering design problem normally involves several engineering disciplines. For example, in a structural design problem, a student might be given the length of a beam and loading conditions, and he is asked to select a beam that can perform well under the distributed loads. This design problem is basic in the complex structural design problems, and often constitutes an element of many engineering design problems. However, it requires aspects of beam section properties and allowable stress to be solved.

Also, each discipline or principle, as a part of the complete design problem, is related to others. That is to say, every element of a design problem contains, and is contained within, another related element of that design problem. Every part of a design problem (and subsequently its solution) is affected by other parts of the problem.

### 3.1.3 Iterative Solutions

The solution process of engineering design problems is often iterative. Most real-world design problems cannot be solved through the use of a simple step-by-step method. Iteration is nearly always required, to achieve the best solution. The most prominent procedures are as follows: pick a system according to the design experience, check if that system satisfied the requirements of the problem. If no requirement is violated, this
system can be regarded as feasible. Otherwise, some modifications have to be made to the system and new design iteration is initialized. Repeat this procedure, until a system, which satisfies all the requirements, is identified.

This feature of engineering design problems determines that, even for the same problem, different designers may have completely different solutions, both of which can satisfy the requirements. It's even possible that they apply different principles or disciplines in the solution process, but the same goal is reached! This flexibility implies that multiple possible solutions may exist for the same design problems. Unique solutions are determined by the properties of the design problems and requirements for the solution method.

3.1.4 Multiple Problem Types

Engineering problems are usually classified into three categories:

- The problem statement gives a system and a set of conditions and the student is asked to evaluate whether the system will perform well in such conditions. For example, in beam design, given the beam section and loading conditions, the student might be asked to check whether the beam is able to sustain the loads. The solution to this category of problems is usually unique.

- The problem statement gives a system and the student is asked to determine the range of conditions where the system will perform well. This category of problem is usually solved by trial and error, and multiple solutions may exist. For example,
given the beam section, determine the maximum load the beam can sustain. This result gives the range of the loads a beam can sustain.

- The problem statement gives a condition, and the student is asked to design a system, so it can perform satisfactorily in this condition. For example, given the loading condition, design a beam section, so the beam can sustain the loads. Usually, many beams may exist, which satisfy the requirements. So the solution is not unique.

EDT satisfies all the features relevant to engineering design problems, thus, it can be used as a general framework for several engineering domains. EDT has employed the Standards, Analysis, Synthesis and Expression (SASE) Methodology to represent design standards in civil engineering design domain, thus it has acquired the characteristics of clarity and ease of use. SASE has the ability to establish a complex structure to represent the relationship among various disciplines. For each design problem, EDT defined a way to find out a specific path called the shortest path from multiple possible solutions. EDT also developed the functions to enable the generation of different engineering problem categories.

3.2 Methodology of the EDT

For the common engineering design problems, sets of design procedures have been developed to obtain the solution. The objective of the EDT is to carry out the design procedures in reverse. To better understand the methodology for problem generation, an example is provided.
Problem Statement: Select the lightest beam supporting distributed load 0.467 kips/ft on a beam with a span of 96 ft. Assume A36 steel and full lateral support.

Solution Technique:

As long as we get the section modulus of the beam, we can get the beam section configuration. The section modulus of the beam can be calculated from the formula:

\[ Z = \frac{M_p}{F_y} \]

\( F_y \): yield stress of steel = 50ksi

\( M_p \): Maximum plastic moment sustained by the beam

For distributed load, \( M_p \) can be calculated as:

\[ M_p = \frac{1}{8} \times q \times l^2 \]

\( q \): distributed load = 0.467 kips/ft

\( l \): length of the beam = 96 ft

so \( M_p = \frac{1}{8} \times 0.467 \times 96^2 = 537.98 \text{kips} \cdot \text{ft} \)

and required \( Z = \frac{M_p}{F_y} = \frac{537.98}{50 \times 12} = 0.897 \text{ft}^2 \)

From all the sections with \( Z \geq 0.897 \text{ft}^2 \), pick the lightest one, and it will satisfy the requirement.
To generate such a problem, the EDT must consider the reverse of the solution technique. Suppose the instructor wants to generate a problem to test the concept of the plastic moment capacity of beams. \( M_p \) can be decided from \( Z \) and \( F_y \). \( F_y \) is a constant, and \( Z \) is the value sought in the problem. Therefore \( M_p \) must be known. Since \( M_p = \frac{1}{8}q \times l^2 \), \( q \times l^2 \) must be known. It is thus necessary that some relationship between \( q \) and \( l \) exists. These constraints between variables constitute a Constraint Satisfaction Problem (CSP). Solution to CSP gives the value ranges of variables which are needed to be plugged into the problem description.

3.3 The EDT Architecture

Fig. 3.1 Structure of EDT

Fig. 3.1 provides an overview of the EDT. The EDT supports two kinds of users: instructors and students. Instructors would use it to generate enough problems for in-class
practice and after-class homework. Students would use it to generate problems for practice, according to their own achievement level, and to improve or consolidate what they have learned in class. The system structure is governed by the following two principles: the students should be able to access the system from a wide range of computer platforms, and the system must support allowing the instructor to develop additional material.

Based on these requirements, the system is developed with open communication standards and the ability to operate on various platforms. The system will utilize the standard WWW protocols and use the standard WWW browser to present the problem and the feedback to the student. The majority of processing is done in the problem generator component, which incorporates a WWW server to distribute materials over the Internet.

The EDT is composed of two main components: the domain knowledge component and problem generator component.

3.3.1 The Domain Knowledge Component

The Domain Knowledge Component contains all the information about the requirements prescribed in a standard of the discipline and their applicability conditions. The domain knowledge component provides a domain representation and a management interface that would allow the instructor to author, modify and maintain information.

The domain representation is a computer representation designed to capture and represent all possible variables and decisions in a topic covered by problems in a problem
set. Each problem set covers a limited set of topics. For example, designs of beam or column are regarded as separate problem sets. SASE is used for the representation.

Domain Knowledge Management UI allows the instructor to create and manage the domain representation.

3.3.2 The Problem Generator Component

The problem generator component consists of five sub-components:

- **Domain knowledge interface:** Users select the specific design concept to be tested through the domain knowledge interface. This sub-component identifies the design rules and all the corresponding applicable conditions. Then a set of design procedures is generated by the domain knowledge interface sub-component, which forms the basis on which any problem on the specified design concept can be solved.

- **Variable identifier:** The variable identifier component obtains from a domain specific database all the variables that participate in the design procedures, with the feasible range of values for those variables. Users can also change the values according to their design.

- **CSP formulator:** This sub component is used to formulate a constraint satisfaction problem from the selected design rules and feasible variable values obtained from the previous two sub components. An external constraint solver is provided to solve the CSP.
• Application specific post-processor: This sub-component translates the solution of the CSP to a problem description. A problem description is formulated by plugging in values for different variables from the CSP solution to a standard problem template.

• Solution generator: This sub component generates the solution procedure and displays it in a readable format. The solution can be stored and retrieved from the problem database.

3.4 Limitations of EDT

The original implementation of EDT has established the basic frame of the engineering problem generator. However, its development only accomplished the basic functionality, without considering the criteria for a good problem generator, mentioned in Chapter 2. The main limitations of EDT include:

• The domain knowledge interface generates a set of design problems without guaranteeing completeness and correctness of the problem.

• EDT uses a limited and un-expandable set of problem statement for converting its results into readable problem description

• EDT provides no support for automated generation of problem-sets.

It is necessary to extend the EDT to account for the disadvantages without sacrificing the functionalities described above. The work described in this thesis deals with the improvements made to the problem generation component of EDT aimed at addressing these limitations.
Chapter 4

Problem Generator System in EDT

4.1 Problem Generation in EDT

This chapter describes the new implementation of the problem generation component of the EDT system. This system expands the capabilities of the previous EDT system by paying special attention to the issues of correctness, completeness, user-friendliness and interaction.

The problem generator component of EDT was designed as a general system capable of generating problems for multiple domains. This design decision was influenced by the architecture of expert systems. Expert systems consist of a generic reasoning shell capable of processing rules/knowledge from a variety of domains. The problem generator component of EDT is designed to serve as a generic problem generation shell capable of processing problem-solving knowledge from a variety of domains. As a result, the successful implementation of the problem generation requires both the definition of the generic problem generation process and the selections of the suitable domain knowledge representation. The requirements on the domain representation can be placed into two broad categories: compatibility with the problem generation process and compatibility with a range of problem domains.

The EDT system uses the SASE representation first proposed by Fenves et.al.[8] to model the problem solving process of engineering design. The SASE representation was developed to support computer manipulation of civil engineering design standards and has been applied to several other domains. Because the choice of representation has
significant impact on the processing of the problem generation component, the next section provides an overview of the representation used by SASE.

4.2 SASE Knowledge Representation

The SASE knowledge representation consists of two major representation elements: the data item and decision table. While the SASE representation deals specifically with issues of representation of a prescriptive design standard it can be applied to represent any design procedure since a prescriptive design standard can be considered a codification of a set of widely acceptable design procedures.

4.2.1 Data Item

A data item or datum is a precise identification of an information element occurring in a standard. The status (satisfied or violated) of each requirement is represented by a datum. Each result or variable generated by a determination is a datum. In addition, every other variable referred to in a standard but not explicitly assigned value by some provision is a datum. For example, the density of a material may be referred to, but not defined, in a standard. Such data are called input data; their values are not determined by the standard itself. All data assigned values by provisions of the standard are called derived data. The logic expressed by a design requirement and its applicable conditions to assign a value to a derived data item is termed as a design rule. Because each requirement has a set of applicability conditions, different design rules may exist for assigning values to a derived data item.
The set of data items plus the expressions of the rules for evaluating and relating the data items contain all the information necessary to evaluate compliance with a standard.

4.2.2 Decision Tables

A decision table defines a set of rules specifying certain actions to be executed based on a specific set of conditions. Decision tables are a convenient means to express the logic for a set of decisions.

A decision table is composed of conditions, actions, and rules. A condition is a logical statement that may have only one of two values: true or false. An action in a general sense is any operation, e.g., it may be the assignment of a value to a variable by means of a formula, or a statement that prescribes a set of conditions in order that a specified set of actions can be performed. A decision table is a structure for defining a set of related rules. The conventional structure of a decision table is shown in the Table 3.1.

```

<table>
<thead>
<tr>
<th>Condition stub</th>
<th>Condition entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action stub</td>
<td>Action entry</td>
</tr>
</tbody>
</table>
```

Table 4.1 Regions of a Decision Table

Each decision table is divided into four quadrants (in this representation, by the double lines). The upper left quadrant is the condition stub defining all logical conditions
that have a bearing on the actions. The lower left quadrant is the action stub defining all possible actions. The upper right quadrant is the condition entry and the lower right quadrant is the action entry. The condition entry lists the combinations of values of the conditions, one set to a column. Each column in the condition entry defines a rule. The action entry indicates which actions are to be executed for each rule. The rule is a logical AND function, that is, the rule is not satisfied unless each of the condition entries it contains is matched.

A simple nomenclature is used in the decision tables. A “T” or “Y” in a condition entry indicates that the condition must be true for the particular rule to apply. An “F” or “N” in a condition entry indicates the condition is immaterial; it can be either true or false. An “X” in an action entry indicates which action is to be taken for a given rule.

Each datum reference in a condition stub is an ingredient of the datum defined by the action. The latter datum is termed a dependent of each of its ingredients.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lpw &lt; Ldaw</td>
<td>F</td>
</tr>
<tr>
<td>Ldaw &lt; Lrw</td>
<td>T</td>
</tr>
</tbody>
</table>

\[ \text{Rnw} = \text{Mnth} \]
\[ \text{Rnw} = \text{Rp} - (\text{Rp-Rc}) \times \frac{(Ldaw-Lpw)/Lrw-Lpw)}{Lrw-Lpw} \]
\[ \text{Rnw} = \text{Rcr} \]

<table>
<thead>
<tr>
<th>Condition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2 Example of Decision Table

Table 3.2 illustrates most of this decision table nomenclature. In this table, rule 1 applies if condition 1 is false since condition 2 is then immaterial; action 1 is taken. Ingredient datum \(L_{pw}, L_{daw}\) need to be known in order to determine that rule 1 applies and to evaluate the dependent datum. If condition 1 is true and condition 2 is also true, rule 2 applies, and action 2 is taken. If condition 2 is false, rule 3 applies and action 3 is taken.
Ingredient data items $L_{pw}$, $L_{daw}$, $L_{rw}$ had to be known to determine which of the rules applies and to evaluate the dependence datum.

4.2.3 Information Network

To provide a more complete representation of the design standard the SASE knowledge representation introduced a concept of information network. An information network is used to represent the precedence relations among the data items in the standard. Because of the structure of the decision tables, the information network is not explicitly represented as part of the knowledge representation but can be computed based on the knowledge stored in the decision tables and data items defined for a domain. The information network is represented as a set of interconnected nodes. The connections are referred to as branches of the network. A branch may only be connected to two nodes, one at each end, and branches may only be connected at nodes, although they may cross over one another at points without impairing the generality of the definition.

Each datum corresponds to a node in the network. These nodes may be: numerical quantities such as material strength; qualitative values such as the type of occupancy of a building; or Boolean values such as the status of a requirement (satisfied or unsatisfied). In order to provide a more concise representation of the relations among data items, each detailed decision table or function is abstracted into a sub-network, or sub-tree, consisting of:

- A node representing the derived item generated by the table or function;
• Nodes representing all the data items occurring in the table or function, i.e., the ingredients; and

• Directed branches from each ingredient to the node representing the derived data item.

The nodes in set (2) are called the (direct) ingredients of node (1); conversely, node (1) is called a (direct) dependent of all nodes in set (2).

Each decision table establishes the value for one data item, so each decision table is uniquely associated with a node in the network. Some nodes represent items that have their values established by a function; in such cases the function is associated with the node as if it were a decision table.

There will be data items that have no procedure for evaluation contained within the standard. The nodes for these items are called input nodes because their values must be supplied by sources of information outside the standard.

Fig. 4.1 shows an example of information network. Each network has at least one terminal node. A terminal node is a root node of a network and it represents a derived variable that does not participate in any other decision table. The terminal nodes of the tree represent the sub-divided segments of the standard, where each segment pertains to a particular type of problem. For example, in structural steel design, the terminal nodes may represent beam or column design.
Fig. 4.1 Information Network

All the derived data items found in the conditions and actions in the decision table are represented by nodes, such as $M_{altb}$, $L_p$, $L_r$, $M_a$, $M_{cr}$. These nodes formulate a sub-tree. It should be noted that only derived data items are represented by a node. Since each decision table establishes the value for a derived data item, each node in the network also represents a decision table in the domain.

According to the same logic, we can build sub-trees for each leaf in the sub-tree of $M_{altb}$. Also $M_{altb}$ could be the leaf of the sub-tree starting from node "Beam". A complete network for a domain can be assembled by interconnecting these sub-trees.

4.3 EDT Knowledge Representation

The EDT uses the SASE representation to manage the design procedure knowledge that forms the basis of the problem generation process. Since the SASE representation was not directly designed to support the problem generation activities, it has been extended with two additional concepts: the problem tree and the solution path.
4.3.1 Problem Tree

Previously, the information contained in the information network has been used to verify the completeness of the information stored in the decision tables and data items. Since the logic of the individual decision tables combined with the precedence relationships represented in the information network consist of the prescribed design procedure indicated in the code, the same representation could be used to represent the required solution procedure for a design problem.

The EDT system uses an enhanced version of the information network to define a concept of a problem tree. The original nodes in interaction network, which represent the derived item node, remain the same. The problem tree adds a new kind of nodes, representing actions associated with a derived node, and connected to its corresponding derived node.

For example, the decision table “Beam” defines two possible actions. Thus, two action nodes are connected to the derived node “Beam”. Furthermore, each action node represents expressions that contain other derived items as ingredients. The problem tree is further extended to show the dependency between the action nodes and their derived data items. In order to distinguish these two kinds of relationships, two types of branches are used. In this thesis, the solid lines connect the action nodes to its parent-derived node and dash lines connect action nodes to its children-derived node.

The problem tree extends the information network representation by clearly identifying the number of possible expressions that could be used to assign the values to a data item and the dependency between these expressions and other data items in the
information network. The benefit of this approach is that it clearly defines the steps necessary to assign a *specific* value to a data item.

![Problem Tree](image)

*Fig. 4.2 Problem Tree*

### 4.3.2 Solution Path

A solution path represents a unique procedure that could be used to assign a value to the root node in the problem tree. A solution path can be obtained by traversing the problem tree starting at the root. Since each node may have more than one action associated with it, the only decision that needs to be made is which one of the mutually exclusive actions should be part of the solution. If the values of input datum are known, the choice of the action is determined by values of the conditions and rule entries in the corresponding decision table. Since these conditions may require values of their ingredients, this process is continued recursively. The collection of nodes needed to evaluate the root of the problems tree and their corresponding actions constitute a
solution path. This represents the correct procedure for solving the design problem for a given set of input data items. Figure 4.3 shows an example of a solution path.

![Solution Path Diagram]

**Fig 4.3 Solution Path**

In the case of problem generation, the values of input variables are unconstrained. In this case it is not possible to find one unique solution path. Rather it is feasible to identify a set of all possible solutions paths that would allow for a value to be assigned to the root node. This creates an additional challenge in developing an automated procedure for problem generation since the selection of the solution path must satisfy the requirements for correctness and completeness.

**4.4 Automated Problem Generation in EDT**

The previous implementation of EDT used the problem generation module to generate a single problem from the specification provided by the user. This work extends
this and makes the following four major improvement to the problem generation component of EDT:

- The problem generator component considers all possible solution paths and automatically selects the most complete and correct option.

- The problem generator component allows the user to specify a preference that would be used in choosing between two equally complete and correct solution paths.

- The problem generation component is extended with a template of problem statements that automates the translation of the selection solution path into a readable problem statement.

- The problem generation component allows for generation of a set a problem all using the same solution path.

4.4.1 Problem Generation and Selection

The first step in the problem generation process is to read in the domain data from the decision tables and to build a corresponding problem tree or trees. The next step is to identify a solution path from the problem tree that includes all the actions specified by users. These actions represent the concepts that the resulting problem must illustrate.

As was shown above, starting with an unconstrained set of input variables usually results in more than one potential solution path. Since they include all the design concepts, all these solution paths satisfy the minimum requirement for completeness. But
there must be one that best matches the subject matter, so some approach must be used to find the most appropriate path. From the point of correctness and completeness, the shortest solution path that includes all required actions can be considered "best" path. This path produces the most focused problem that includes the minimum amount of other concepts. Therefore the purpose of this problem is made clear.

The previous implementation attempted to accomplish this by performing the breadth-first search of the problem tree to find the first path that includes all required actions.

The following illustrates this approach:

![Diagram](image)

**Fig. 4.4 Previous Implementation**

Figure 4.4 shows a problem tree. Assume the user specifies the following actions: A2 of Beam, A1 of V_n. Two action nodes provide two ways to calculate the value of root node, but since A2 is selected, the solution path must go through A2. There are three
ingredients of this action node, \( M_p, M_{altb}, \) and \( V_n \). and all of them must be evaluated. For \( V_n \), any action can be chosen so the algorithm picks \( A1 \). Since this action node doesn’t have any ingredients the algorithm returns to other nodes. For \( M_p \) and \( M_{altb} \), no action has been specified, so again, according to the breadth-first strategy, the first actions, \( A1 \) and \( A3 \) are taken. This continues until all selected branches reach nodes with no new ingredients.

4.4.2 Current Implementation

The previous implementation, while sufficient, has two major problems: the efficiency involved in generating the problem tree for every search and the possibility of not finding the “best” solution path. The new approach addresses both of these issues.

In this approach, the problem tree is established only once at system start-up. After the problem tree is generated and before the user selects the desired actions, the system identifies all feasible solution paths through the problem tree and sorts them from the shortest to the longest. In order to distinguish between two paths of similar length we introduce a new concept of weight of the path that combines the notion of path length with the preference criteria for the actions associated with that path. The concept of weights used in this system is explained below. After all the weights of the feasible solution paths are obtained, they are compared and the paths are ordered according to the weights, from the smallest to the largest. Even in cases when two paths are the same length, they can be distinguished according to the total weight. Then after the users have defined the actions, the system will check these ordered paths, from the shortest one to
the longest, and find out the first path that includes all the required actions. This path is
the shortest one of all the paths that includes all required actions.

4.4.3 Solution Path Weight

We define the weight of each path as the sum of the length of the path and the
summation of weight of each node. The weight of the node represents the relative
importance of multiple actions for a table. Because the algorithm is free to choose any
action in generating a solution path, the weight allows the instructor to bias the selection
of the solution to include actions that are most common in typical design cases. This
ensures that problems closely mirror common design situations.

In assigning numeric weights, it is assumed that the length of the path plays more
importance than the weights of individual actions. This guarantees that the weight is only
used to discriminate between paths of equal lengths and that a shorter path is always
preferred to the longer one, independent of the weights of their respective nodes. The
procedure requires assigning a length multiplier $a$ to all paths such that the sum of all
node weights $b_i$ should be less than $a$.

Fig. 4.5 illustrates an example. The relative weight of each node is labeled inside each
node. The length multiplier is set as 100 for all paths.
4.4.4 Templates

An important step in problem generation is to convert the solution path into the readable problem statement. The system eases this step by providing a user-defined database of standard problem templates that can be used to formulate a problem statement. It is important to note that a problem description depends on the notation and representation used in specific domain. For example, in structural beam design problem, if the breadth of the flange, width of the flange, depth of the beam and dimensions of the web are the input variables in the solution path, the problem descriptions for a structural beam are stated in terms of the section designation. In this implementation a template is composed of five parts:

- Name: each template has a unique name that gives an indication of its content.

- Wording: each template has a HTML template that defines the readable form for a problem statement. HTML \texttt{var} tag is used to indicate the variables that must be
filled in to convert the template into the problem description. The use of HTML allows the problem description to include a mix of text, drawings and other media.

- **Expression**: a list of expression that relate the variables defined in the wording field and the input variables in the domain. These expressions define the logic used to map from problem description to the variables used to solve the problem.

- **Variables**: defines the list of input variables in the problem. This list is computed from the template and the expressions fields and allows EDT to automatically match the solution path and its input variables to a problem description that uses the same set of input variables.

- **Type**: defines the class of problem association with the template. For example, in the domain of structural design, there are three kinds of problems that are common: check of a given section for a load, design of a section of a given load, and calculation of a maximum allowed load for a given sections. These are important distinctions and they are captured in the contents of this field.

Table 4.3 shows an example of a template for a check type problem description.
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>concen_beam</td>
</tr>
<tr>
<td>Wording</td>
<td>A &lt;var&gt;section&lt;/var&gt; beam supports a concentrated load of &lt;var&gt;p&lt;/var&gt; on a simple span of 20'. A36 steel and full lateral support. Consider moment only. Check the section. &lt;br&gt; &lt;center&gt; &lt;img src=<a href="http://images/problems/two.gif%3E">http://images/problems/two.gif&gt;</a> &lt;/center&gt;</td>
</tr>
<tr>
<td>Expressions</td>
<td>p = V\text{max}; M\text{max} = V\text{max} \times 20 / 2; L_b=0;</td>
</tr>
<tr>
<td>Variables</td>
<td>V\text{max}</td>
</tr>
<tr>
<td>Type</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.3 Example of a Problem Template

The templates are organized based on the domain and are further classified based on the problem type.

Because the templates may have several equations that define the relationship between the problem description and the solution path, these equations have to be considered when values are assigned to all variables in the solutions path. The approach taken in EDT is to add these equations as additional constraints to CSP problem that is formulated from the solutions.
The current implementation of EDT includes a relational database that stores the defined problem descriptions and a web interface that allows the instructor to add, delete and modify problem templates.

4.4.5 Multi-Problem Generation

The EDT uses the selected solution path and problem template to formulate a CSP problem. The solution of this problem will provide the values of input variables in the solution path that would force the solution to a problem to follow the design procedure defined in the solution path. EDT has chosen to use an interval CSP that produces a range of feasible values for each variable in the solutions. This allows the EDT to automatically generate more than one problem for a selected solution path. This is critical in allowing EDT to generate a set of different problems that all test the same concepts with the same level of complexity.

This approach presents significant benefits but also requires further processing to select the values for design variables that would be used in formulating the problem statement. EDT provides two ways of dealing this additional step:

- an approach that allow the user greater control in choosing the problem variables, but that requires problems to be generated one at a time.

- an approach to allow EDT to generate a set of problems automatically.
4.4.6 User-Guided Generation of Problems

In this process, all the basic variables, which appear in the template, are displayed to the user. Associated with each variable is its lower and upper bound of value ranges as determined by the EDT, and a default number of intervals to divide the range into. Users can define new ranges and interval numbers to each variable, or use the default values. Every time a variable is selected, its interval is subdivided according to \( n \) points where \( n \) is defined by the user (Fig. 4.6). The instructor can choose any of these values, which is then added to the CSP as a new constraint. The CSP uses the newly augmented constraint set to compute new intervals for the other un-constrained input variables. Different values for each variable would produce different value ranges for other basic variables in the solution path.

![Diagram](image)

Fig. 4.6 Intermediate Point

If the instructor is not satisfied with any of the basic variable value selected or if the CPS fails to find a solution at any point in the process, the instructor can backtrack to select different values for input variables.

This approach allows the instructor to provide strict control over the values of all input variables, but simplifies the problem generation process by automating all other stages of this process. Using this approach the instructor can quickly generate a set of similar but unique problems to use in class.
4.4.7 Automated Generation of Problems

EDT also provides an automated mechanism for generating problem sets. The instructor is expected to only specify the number of problems that should be generated and define the ratio point that would be used to choose the values for the input variables.

EDT uses the first variable in the variable list to drive the problem generation process. The interval for this variable is subdivided based on the number of problems requested by the instructor. Each of these values becomes a starting point for generating a problem statement. The value of the variable is added to the CSP that is solved to produce new intervals for the remaining input variables. The ratio point is used to determine the values for the rest of the input variables (Fig. 4.7). An input variable is chosen at random and the ratio point is used to assign its value by selecting a value from its current interval. This process is repeated until all variables have a value. The resulting set of values defines a single problem in the problem set.

This procedure is repeated for each value produced by subdividing the initial input variable. The result is an automatically generated set of problems all following the same solution path with unique values for all input variables.

![Fig. 4.7 Ratio Point](image)
4.5 Summary

This chapter described several major modifications to the problem generation strategy in EDT. The motivation for these modifications was the need to ensure the efficiency, correctness, completeness and user-friendliness of the problem generation process. The next chapter provides a case study of the operations of the problems generation of EDT and highlights the UI improvements that were implemented to ensure the user-friendliness of this system.
Chapter 5
Case Study

In this chapter the operation of the new problem generator for steel member design will be illustrated by generating an example problem that will be used to test the concept of plastic moment capacity of beams under distributed load.

As described before, when the system starts up, a problem tree is generated from all the decision tables stored in the database. Then all the possible solution paths are identified in the tree and sorted according to the weight method. The next stage is problem generation that involves the following six steps: problem definition, variable definition, constraint output, template specification, section selection and solution key.

The problem definition step involves the specification of initial parameters like the problem domain and the set of concepts that must be included in the problem.

![Problem Definition](image)

Fig 5.1 Problem Definition
Fig 5.1 shows the window where the concepts to be tested are defined. All decision
tables defining the domain are displayed in the window on the bottom left. They are
organized in the format of a tree. Each node of the tree represents a decision table, and
each leaf of the node represents an action in the decision table. Since each decision table
contains at least one action, each node must have at least one leaf. When a node is chosen,
the text representation of the decision table is displayed in the text-area on the top. When
a leaf is chosen, the corresponding action can be added to the list of actions on the right
of the window, by clicking on the Select button. The user can remove an action from the
list at any time by highlighting the action in the list and clicking on Delete. Finally,
clicking on the button Done closes the window, saves all changes and the selected actions
will be input into the system for parsing.

The design example considered in this chapter belongs to the “Beam” problem
domain, so the decision table corresponding to the domain of “Beam” is selected. There
are two possible actions in this table. The action corresponding to “Beam=1” means that
the beam is capable of carrying the assigned load, while “Beam=0” means the opposite.
In the example problem, the plastic moment capacity of the beam is not to be exceeded
and hence the “Beam=1” action is selected. In addition, user needs to indicate that the
concept of plastic moment capacity should be included in the problem and the
corresponding decision table $M_p$ is selected. There are two possible actions corresponding
to $M_p$, which represent two ways to calculate plastic moment capacity of beams.
Assuming the user wants to test the concept of specified minimum yield moment, so
$F_y*Z$ is selected.
After the necessary actions are specified, all the solution paths in the problem tree are searched. Since they are sorted according to the length, from the shortest one to the longest one, the first path that includes all the selected actions is chosen as the “best” solution. Once the solution path is obtained, the user needs to specify feasible ranges for the variables in the domain. Thus the next stage in the problem generation process is “Variable Definition”.

**Fig 5.2 Variables**

In order to avoid the redundant work involved in generating different variable list for different solution path, a complete list that includes all the basic variables available in the problem domain is provided to the user. Fig 5.2 displays the list of parameters. The program is designed so as to let the user perform the design with the minimum amount of data entry. The default ranges for data values are stored in the database and can be loaded and displayed in the data boxes beside each variable. Users can modify any or all of the
values. To modify the existing variable ranges, the user enters the updated data into the data boxes and when finished, and clicks the "Next" button. The revised list of variable ranges is then saved.

The actions defined in the first step, the selected solution path, and the variable value ranges obtained in the second step form a constraint satisfaction problem. An external constraint solver is used to solve the CSP. The results are displayed in the next step "Constraint Output" step.

Fig 5.3 Constraint Output

The window in Fig 5.3 is quite similar to Fig 5.2, but includes only variables obtained from the solution to the CSP. They are the variables found in the expression that form the solution path. The new value ranges are obtained from the interval constraint solver's solution of CSP. Again, the user can modify the provided value ranges.
In this step, the user has three choices:

- "Another": If the user is not satisfied with the solution path found in step one, the next shortest path can be chosen simply by clicking the "Another" button.

- "Recheck": This allows the user to change the ranges for the values in the form and to perform constraint satisfaction to obtain new ranges for the variables in the path.

- "Next": This allows the user to proceed to the next step in the problem generation process.

Fig 5.4 Templates

After the basic information about the design concept has been loaded, step 4 (Fig 5.4) is referred to as "Template Specification" and mechanism for selecting a problem
generation approach. The window is divided into three main sections, **Template List**, **Template Output** and **Options**.

**Template List** shows a tree, which displays a list of possible templates. These templates are categorized and organized according to the method described in chapter 4. When one of the templates is selected, the system connects to the database and finds the corresponding problem statement and equations associated with the template, and displays them in the text area on the bottom, in the next step called as **Template Output**.

In the example problem, since a section design problem for distributed load is to be generated, the user selects “design section” node and “distributed load” leaf of the node.

In **Options** section the user specifies the input according to either of the two problem generation methods introduced in Chapter 4: multi-generation and single-generation. For multi-generation, two inputs are needed. The first entry is the number of problems the user would like to generate. The second entry is a drop-down menu that shows all available criteria. The default value is “0.25”.

In the example case, it is assumed that, the user wants to generate ten problems, so 10 is input in the first text field. 0.25 is chosen as the ratio point of the range for all the variables.

The constraints from the selected template are added to the original constraint satisfaction problem and require additional step of re-solving the constraint satisfaction problems. The CSP is solved 10 times, once for each value of the selected variable. The solution to each problem is combined with the template to represent a unique problem.
Select the lightest beam supporting distributed load 0.467529296875 on the span of 96.0. Assume A36 steel and full lateral support. Consider moment only.

Fig 5.5 Sections

In step 5 – “Select Section”, the indices of the problems are displayed in a scroll pane for the multiple problems generated. Each of them corresponds to a problem description. When a corresponding index is chosen, the problem description is shown in the text area.

After finishing the generation, the user can request a solution to the problem. A summary of final solution is presented in the solution panel (Fig 5.6). It displays all the necessary and useful information, including the problem generated, each step of the solution procedure, and the crucial variable at each step. Other formats of the solution such as PostScript file are also available.
Fig 5.6 Solution Key

The user can save the problem to a problem database. A unique ID is assigned to each problem, so that the problem can be retrieved for later use (Fig 5.7).
Chapter 6

System Performance

EDT has been designed as an intelligent assistant to academic instructors and students. To verify that EDT has achieved its intended goal, its performance is tested according to the specific criteria mentioned in Chapter 2.

6.1 Correctness

EDT uses SASE methodology to represent all the domain knowledge needed to generate a problem. The SASE methodology offers several mechanisms to ensure correctness of the domain knowledge. First, the SASE methodology provides an automated mechanism that ensures that each decision table is correct by ensuring that an action is provided for all possible values of all conditions. Also there is a precise definition of the input that the system could accept and the output that should result from the inputs.

In addition, in EDT, the user can only choose from the values provided by the system, or select values from the default value ranges stored in the system. The problem is generated and formulated using the results of CSP and the pre-defined template stored in the system. The templates are carefully designed to closely match the concepts being tested and employ the effective wording. This ensures the EDT system is designed fully that the generated problems represent the desired knowledge and skills, and guarantees the correctness of the result.
6.2 Completeness

All the design concepts in the domain are stored in the database and available for retrieval. The detailed content of each design rule is represented by a decision table structure, and the hierarchy of design rules is displayed in the format of a tree. This guarantees that users can select all the design rules that must be included in the problem. In the current version of EDT, the problem generated according to the shortest path algorithm is regarded as the most complete problem.

6.3 User-friendliness

The friendliness and the effectiveness of the user interface are evaluated by two criteria: user input and replay capability. These are described below.

6.3.1 User Input

Since problem generator requests input from the user during the problem generation process, the manner in which the users are prompted and their response is handled is crucial to the system's overall success. The input formats used in the problem generator include keystrokes, function keys and mouse action.

In order to make the system friendly to the users who are not skilled in the use of the keyboard, most input formats of the system allow the user to select an item by clicking on its label or icon, or by selecting its number. The user is not required to do significant amount of typing to interact with the system. However, in some situations where keystrokes are required, default inputs are provided, so the user only needs to alter part of the input rather than entering all values. This significantly reduces the amount of typing.
6.3.2 Replay Capability

A second feature in the user interface is its ability to "replay" or "rerun" a particular step without having to restart the entire process. Typically, this feature is invoked when the user wants to rerun a generation step after making a change to the generated intermediate result. This allows them to expose the "what-if" scenarios and explore how a change to a variable may impact the problem generation process.

6.4 Interaction

System status is displayed in the status bar available in the interface at each step in the problem generation process. The user is informed whether a process step is processing, done, in an error mode, or waiting for the input. When the processing time is long, feedback on the status is particularly important.

With a legal input, a message such as "generating feasible paths", "generating feasible ranges" is provided as feedback that acknowledges acceptance of the input and indicates that the system is waiting for the process to complete. Dynamic progress indicators like progress bars are used to provide feedback on how long the waiting might be, and how much of the execution is left.

The system also checks the consistency of the input and automatically defines output features, whenever necessary. For improper or impractical input values, a warning or popup message will be displayed, and the user will be asked to re-enter the input. As an example, when the user proceeds to generate problems without choosing templates for
the problem, the pop-up message will be displayed (Fig 5.1). Also now, additional user interface elements, such as tool tips, are used to provide additional help to the user.

Fig. 6.1 Error Message
Chapter 7

Conclusion

This chapter presents a summary of this work and outlines some possible extensions and improvements.

7.1 Summary

This thesis describes the development, working mechanisms and improvements made to the problem generation component of the EDT system. In the previous implementation of EDT, a problem generation module was used to generate a single problem from the specifications provided by the user. Considering the issues of correctness, completeness, user-friendliness and interaction, the system is expanded by making the following improvements:

- In the previous implementation, a problem tree is generated after the actions have been defined and hence the tree has to be generated for every search. In the current implementation, a new problem generation process is provided, in which the tree is generated when the system starts up and before the actions are specified so that the tree needs to be generated only once. This improves the efficiency of the system.

- Instead of a breadth-first search, a new sorting method based on the weight method is provided, which guarantees the completeness of the system.
• A user-defined database of standard templates is provided to convert the solution path into a readable problem description. This improves the correctness and user-friendliness of the system.

• EDT has been extended to support automatic generation of problem sets or unique problems, that all use the same solution path.

7.2 Limitations of This Work and Future Extensions

Although the new approaches are well designed, several problems remain unresolved in the current implementation, and need to be addressed in the future implementation.

The general EDT framework is designed to be an automated problem and solution generator for routine engineering design. However, the current version was only tested on problems from civil engineering design domain. It’s necessary to test how well it extends to other engineering domains.

The second limitation is that users of the EDT can’t provide feedback to the developers. Feedback helps the developers to maintain and improve the system. By getting the feedback from the users of EDT, developers quickly identify and correct problems with the system as soon as they arise.

The third problem is that the current system doesn’t have a security mechanism. An ideal system would assign different access rights to different levels of users, and allow each user to have a separate account. Included in this would be the ability to track the activities of the users to provide the capacity to evacuate how the capabilities of the system match the user’s requirements.
These concerns remain unresolved in the implementation, and can be addressed as future topics.
References


