

**AUTOMATED STRUCTURAL  
DESIGN USING IMPLICIT ENUMERATION  
TECHNIQUES**



**Judin Bin Abdul Karim**

December 1987

**AUTOMATED STRUCTURAL  
DESIGN USING IMPLICIT ENUMERATION  
TECHNIQUES**

by

**Judin Bin Abdul Karim**

---

A Dissertation Presented to the  
FACULTY OF THE GRADUATE SCHOOL  
UNIVERSITY OF SOUTHERN CALIFORNIA

In Partial Fulfillment of the  
Requirements for the Degree  
DOCTOR OF PHILOSOPHY  
(Civil Engineering)

December 1987

Copyright 1987 Judin Bin Abdul Karim



# Table of Contents

	Page
Acknowledgements .....	ii
List of Tables .....	viii
List of Figures .....	xi
<b>1 Introduction .....</b>	<b>1</b>
1.1 General Remarks .....	1
1.2 Objectives and Scope of Study .....	6
1.3 Organization .....	8
<b>2 Technical Review of Previous Work .....</b>	<b>11</b>
2.1 Terminology and Approaches .....	11
2.1.1 Automated Design .....	11
2.1.2 Structural Optimization .....	14
2.1.3 Optimization Methods .....	15
2.2 Optimality Criterion .....	16
2.2.1 Optimality Criteria Methods .....	17
2.3 Mathematical Programming .....	19
2.3.1 Classical Indirect Methods .....	19
2.3.2 Direct Search Methods .....	20
2.3.3 Linear Programming and Sequence of Linear Programming Formulations .....	24
2.3.4 Unconstrained Minimization Approaches To Constrained Problems .....	27
2.4 Discrete Variables .....	29
2.4.1 Integer Programming: Gomory's Algorithm .....	31
2.4.2 Zero One Integer Programming .....	32
2.4.3 Diagonal Enumeration .....	35
2.4.4 Discrete Variable Direct Non Gradient Search .....	36
2.4.5 Discrete Variable Gradient Based Methods .....	38
2.4.5 Discrete Dynamic Programming .....	39
2.4.6 Segmental Method .....	40
2.4.7 Backtrack Method .....	42
2.5 Approach For Automated Design .....	42

<b>3 Search Strategies .....</b>	<b>46</b>
3.1 Introduction .....	46
3.2 Trees and Graphs:- Terminology .....	46
3.2.1 Graphs .....	46
3.2.2 Trees.....	48
3.3 Tree Search Criteria .....	51
<b>4 General Purpose Methods. ....</b>	<b>52</b>
4.1 Generate and Test and Hill Climbing.....	52
4.2 Breadth First Search.....	53
4.3 Depth First (Backtracking) .....	54
4.4 Best First (Branch and Bound).....	57
<b>5 Development of Search Strategies And Applications .....</b>	<b>59</b>
5.1 Backtrack - Strategy and Applications. ....	59
5.1.1 Example 1: Fifteen member simply supported truss with 3 design variables .....	61
5.1.2 Example 2: Fifteen member simply supported truss with 8 design variables .....	67
5.1.3 Example 3: Welded Plate Girder Design.....	70
5.1.4 Example 4: Reinforced Concrete Beam .....	74
5.2 Branch and Bound - Strategy And Applications.....	82
5.2.1 Example 1: Fifteen member simply supported truss with 3 design variables .....	84
5.2.2 Example 2: Fifteen member simply supported truss with 8 design variables .....	91
5.2.3 Example 3: Welded Plate Girder Design.....	94
5.2.4 Example 4: Reinforced Concrete Beams .....	96
5.3 Extension of Application of Branch and Bound to Plastic Frames .....	101
5.3.1 Example 5: Single Story Single Bay Frame .....	104
5.3.2 Example 6: Double Story Single Bay Frame .....	110
5.3.3 Example 7: Double Story Two Bay Frame.....	116
<b>6 Heuristics and Learning Process .....</b>	<b>122</b>
6.1 Learning Process.....	122
6.1.1 Application of Learning Process.....	125



6.2	Mode Generation.....	126
6.2.1	Inclusion and Utilization of Modes Generated by Randomly Sizing the Member Sizes.....	128
6.2.2	Generation and Utilization of Simple Modes.....	134
6.3	Heuristics .....	137
6.4	Most Promising First-Utilization of Evaluation Functions.....	138
6.5	Heuristic Applications .....	142
6.5.1	Use of Interpolating Factor For Frame Design Problems.....	142
6.5.2	Results Obtained.....	143
6.6	Plastic Design Problems With Large Number of Available Sections For Each Variable.....	148
6.6.1	Use of Interpolation Factor For Frame Design Problems With A Large Possible Set of Values.....	149
6.7	Use of Initial Guess.....	151
6.7.1	Using Actual Optimal Solution As Initial Guess .....	152
6.7.2	Using A Poor Initial Guess.....	156
6.7.2	Poor Guess versus Optimal Initial Guess.....	157
6.8	Computer Network-Structural Optimization. ....	159
6.8.1	Parallel Process.....	159
6.8.2	Application and Results.....	160
6.9	Kappa Search Strategy: The Best Few Search.....	167
6.9.1	Selecting The Size of Kappa.....	168
6.9.2	Application of Kappa Search Strategy.....	169
6.10	Beta Search Strategy: Storing The Best Few Nodes.....	172
6.11	Biased Search Within A Realistic Range of Possible Values With Beta Distribution .....	175
6.11.1	Realistic Range of Possible Values.....	175
6.11.2	Biased Search Strategy .....	177
6.11.3	Application of Biased Search Strategy.....	179
<b>7</b>	<b>Summary and Conclusions.....</b>	<b>191</b>
7.1	Discussion of Work Done in this Study.....	191
7.2	Topics for Future Investigation .....	200

<b>Appendix A</b> .....	<b>203</b>
A1 Evaluation of Load Factor Using Assumed Collapse Modes .....	203
A2 Illustrative Example .....	205
<b>References</b> .....	<b>211</b>



# 1 Introduction

## 1.1 General Remarks

Automated design is a topic of interest in every engineering discipline. The contributing factor is that design, unlike analysis, is a multivalued problem. In general, most design problems do not have a unique solution.

Early approaches to automatic design of structures focused on the automation of standard office design practices. However, this approach is useful primarily for specific cases only. It lacks flexibility and is difficult to generalize. The second approach is analysis-oriented, and is based on structural criteria. The third approach involves the application of mathematical programming ideas.

Numerous automatic design algorithms have been developed based on these approaches. However, almost all the available algorithms for the optimal automatic design of engineering structures make the assumption that design variables, such as member sizes, are continuous-valued, and that the material and its properties from which each member is made, is known in advance. In practice, however, designers rarely have the freedom to choose member sizes from a continuous range, as this implies a separate and expensive fabrication process for each member. Usually, designers are

restricted to choosing member sizes from a discrete set of available pre-fabricated sizes. Furthermore, although the range of available sizes may be large, it is common practice to pre-select a small subset of perhaps five or six of these sizes which may actually be used in design.

Designers may also find difficulty in specifying the material properties of each member before the design process starts. Typically, a particular steel section, reinforcement bar or concrete may be produced in several steel grades or concrete strengths, each with a different set of material properties and costs. Ideally, the design process itself should select the most appropriate material for each member.

Furthermore, difficulties are encountered with the design requirements and codes specifications. Many of these are developed by designers after several years of experience and successful design. Furthermore, these specifications, as presented in the codes, represent functions that are nonlinear and discontinuous. Thus, they are not easily formulated as a continuous function as required in mathematical and optimal criteria algorithms.

Also, the objective function, in the practical world, is the cost of the structure rather than weight. Cost is a function of several factors, and it is not a continuous function of the design variables; rather, it may be a function of economics (availability



and cost of the materials) and fabrication (construction methods). Thus, the discrete properties of the variables also imply that the objective functions vary in a discrete, discontinuous fashion.

Although the discrete nature of member sizes and design specifications has long been recognized, little research has been devoted to automated structural design incorporating discrete design variables and discontinuous constraints. The characteristic of these problems precludes the application of many widely used mathematical optimization and optimal criterion procedures. For example, linear programming is not directly applicable because of nonlinearity of both the constraints and the objective function. Dynamic programming is not directly applicable because there is not a sequential flow of information. Gradient search and pattern search methods are not applicable, because the objective function is neither continuous nor unimodal and because the design variables are defined only at discrete values.

Attempts have been made to add these discrete properties to existing, computationally efficient, continuous valued optimization methods. An initial approach was to replace the discrete value variable with a continuous one, to solve the resulting optimization problem with a "continuous" method, and then finally to round off the solution to the nearest available size.

This approach, while useful in some problems, is not without pitfalls.

Consider a hypothetical function with contours as shown in Figure 1.1.1. The  $\oplus$ 's represents available combinations of  $d_1$  and  $d_2$ . The optimal solution of the continuous problem is at O ( $x_1, x_2$ ). If the values of  $x_1$  and  $x_2$  obtained for this solution are rounded to the nearest integer values, the result is neither an optimal nor a feasible solution. The actual discrete optimal feasible solution is represented by P, and P is not adjacent to O in each variable.

Thus, even if a continuous solution may be obtained from a continuous design algorithm, there is no way of determining a priori for each structural member whether it should be rounded up or rounded down in size. Ad hoc rounding rules may be devised for some structures, but they offer no guarantee of optimality or feasibility. One section size may in fact separate a good design from an optimal design. Even so, for practical situations, this rounding process turns out to be a combinatorial problem of immense size. Consequently, the use of continuous optimum design with some approximate rounding rules cannot be recommended, especially when the member sizes are limited to a small discrete set covering a wide range of sizes.



Unfortunately, even if there exists a continuous analog of the discrete problem, the complexity of these methods is then greatly increased and its efficiency severely reduced.

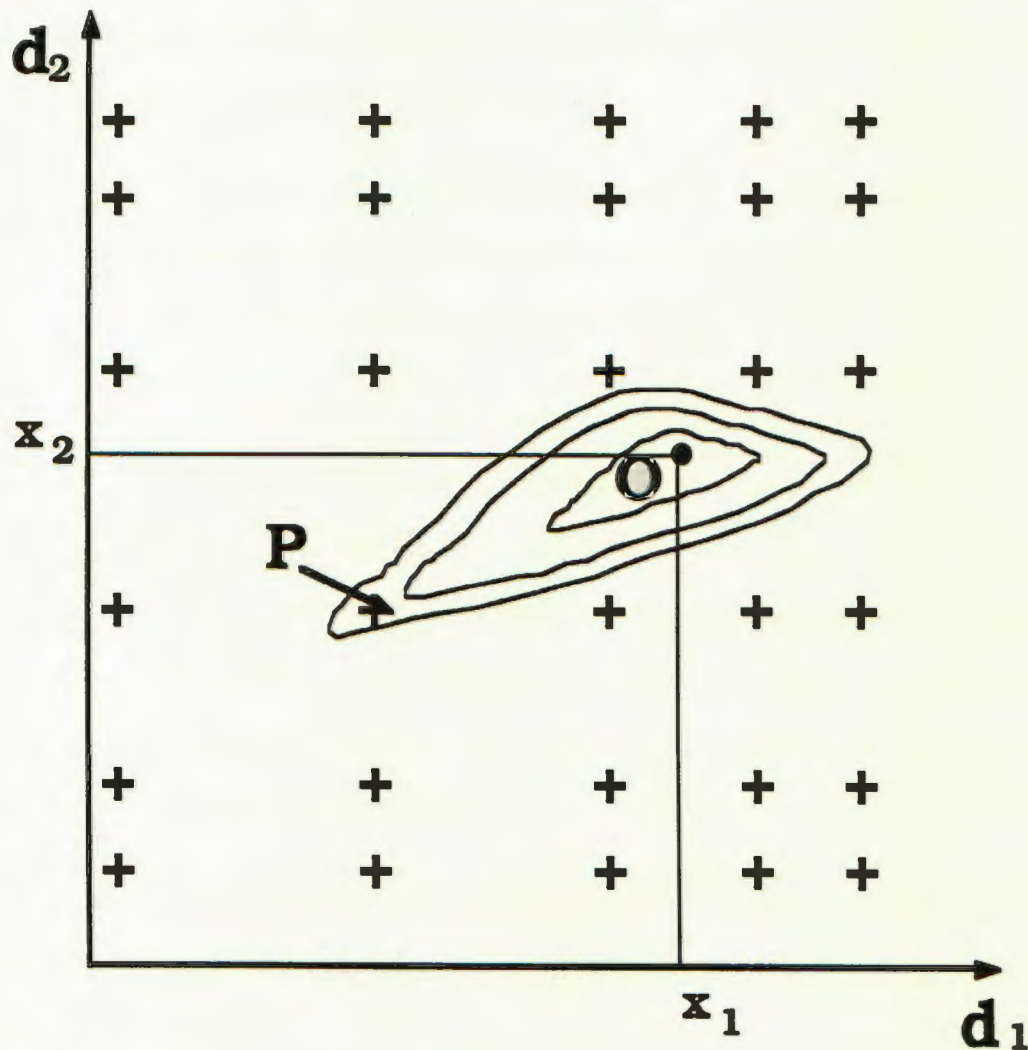


Figure 1.1.1  
Continuous Solution and Discrete Solution