Speciation, Clustering and other Genetic Algorithm Improvements for Structural Topology Optimization

by

James Wallace Duda

B.S., Mechanical Engineering
Case Western Reserve University, 1994

Submitted to the Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Mechanical Engineering

at the

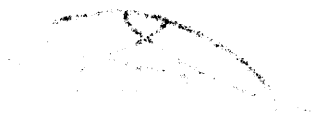
Massachusetts Institute of Technology

May, 1996

Signature of Author _____
James Duda
Department of Mechanical Engineering
May 31, 1996

Certified by _____
Mark J. Jakiela
Associate Professor, Mechanical Engineering
Thesis Supervisor

Accepted by _____
Ain A. Sonin
Chairman, Department Committee on Graduate Students



Speciation, Clustering and other Genetic Algorithm Improvements for Structural Topology Optimization

by

James Wallace Duda

Submitted to the Department of Mechanical Engineering on May 31, 1996
in partial fulfillment of the requirements for the Degree of Master of Science in Mechanical Engineering

ABSTRACT

Genetic algorithms are used to search for optimal structural topologies. Modifications to basic genetic algorithm techniques are implemented to increase computational efficiency, avoid premature convergence to a single solution, and solve new categories of problems. GA's are a search and optimization tool based on the principles of evolution and survival of the fittest. Potential designs are represented by chromosomes, each of which receives a fitness score based on the quality of the design it represents. As in nature, the more fit a chromosome is, the more likely it is to survive and "reproduce," combining with another "parent" chromosome to produce new "child" chromosomes. As this process of evaluation, selection, and reproduction is iterated, the population of chromosomes "evolves," and new and improved designs are generated. In this study, an allowable design domain is discretized into several binary, material/void elements, yielding a combinatorial search space. One chromosome represents one design, which can be evaluated using finite element analysis or analytical techniques. Extending previous efforts with the same basic representation and search technique, this research proposes several methods for improving genetic algorithm performance. New population initialization and parent selection methods are implemented to reduce run-times and decrease the number of poor intermediate designs generated and evaluated by the algorithm. Fitness sharing and speciation are used to distribute subsets of the evolving population over many local optima, preventing premature convergence to a single solution when multiple, equally good solutions exist. The resulting distribution of sub-populations is analogous to different species exploiting different niches in an ecosystem. Statistical cluster analysis techniques are used to divide the population into sub-species and to quantify the extent to which a population is speciated. Additionally, this cluster analysis is used to discourage the mating of dissimilar designs (designs from different clusters). Results show that these modifications to basic genetic algorithm techniques result in shorter run-times and greater diversity of solutions. Finally, a hybrid GA / simulated annealing method is introduced for topology design of adaptive structures.

Thesis Supervisor: Mark J. Jakiela
Title: Associate Professor, Mechanical Engineering

Acknowledgments

I would like to take this opportunity to thank the people who made this work possible. First, I would like to thank Mark Jakiela, my thesis advisor, for his guidance and support. I would also like to thank Colin Chapman, for helping me decipher some old computer code, and Matthew Wall, for helping me write new code. The software developed in this investigation relied on the ever-changing GA library created by Matthew Wall and finite element analysis software written by Ashok Kumar. I'd also like to thank all the members of the MIT CADlab for making the lab a pleasant and sometimes even fun place to work, and for not complaining when my code was slowing down their machines. Finally, I would like to thank the National Science Foundation, whose funding has made this work possible.

Table Of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

This chapter provides a brief overview of the chapters that follow. The motivation for this research is presented, followed by a more specific statement of the problem faced and the approach taken. Finally, the organization of this thesis is outlined.

1.2 Motivation

Design is an iterative process which may be divided into the following four stages: formulation of functional requirements, conceptual design, optimization, and detailing (Kirsch, 1981). Iterations of this process are often required as the initial design frequently is in some way inadequate. The conceptual design phase is a critical step in this process. In it, the general characteristics of the design (shape, topology, material, *etc.*) are defined. Although this stage is very important, relatively few computational tools exist to aid the designer in generating conceptual designs. Once a conceptual design is created, several methods exist for performing some type of optimization. The final stage of the design process, detailing, must generally be performed by an engineer, not a computer-based system. This final phase involves the final definition of the design, usually based on the designer's experience and engineering judgment.

Recently, genetic algorithms ("GA's") have been introduced as a method for generating and optimizing conceptual designs. In particular, this approach has been applied to topology design optimization for load-bearing structures. Such approaches have experienced some degree of success, although certain performance limitations exist. In particular, GA's tend to require a great deal of computation and generally produce a single, "best" conceptual design.

Researchers have also been seeking to discover how and why GA's (and other techniques that simulate evolution) work and how they should best be used (see, *e.g.*, Eshelman,

1995). One subset of these more theoretical efforts addresses techniques for modeling phenomena closely related to the basic concept of simulated evolution, such as co-evolution, emergence, and speciation. As models of these related phenomena become better understood, they will undoubtedly find use in the solution of difficult problems.

This investigation focuses on developing an improved genetic algorithm-based system for the conceptual design optimization of load bearing structures using advanced GA techniques including speciation and clustering. Although structural topology optimization is a very specific design problem, the methods developed in this study are applicable to a variety of problem domains. In fact, one of the strengths of GA's as an optimization technique is their ability to perform optimization in a wide range of problem domains.

1.3 Objective

The general objective of this investigation is to develop an improved system for genetic algorithm-based conceptual design, specifically applied to the problem of structural topology optimization. In particular, the following goals are set forth:

To improve the efficiency of genetic algorithms as a tool for structural topology optimization. This involves reducing GA run-times through the use of better GA operators and methods.

To increase the diversity of solutions produced by a genetic algorithm-based optimization run. It is desired that one run of a genetic algorithm produce several optimal or nearly-optimal designs, allowing the designer to select the "best" design based on fine details and engineering judgment.

To apply this approach to example problems in order to demonstrate its capabilities and measure its performance.

To introduce a method for using genetic algorithms to perform optimal topology design of adaptive structures.

1.4 Problem Definition

Typically, genetic algorithms are applied to structural topology optimization problems posed in the following form:

Given a design domain defining the region in which the structure must exist, one or more loads to be applied to the structure, and a set of support points on the structure (see figure 1.1), generate the structural topology which optimizes performance while satisfying a set of constraints.

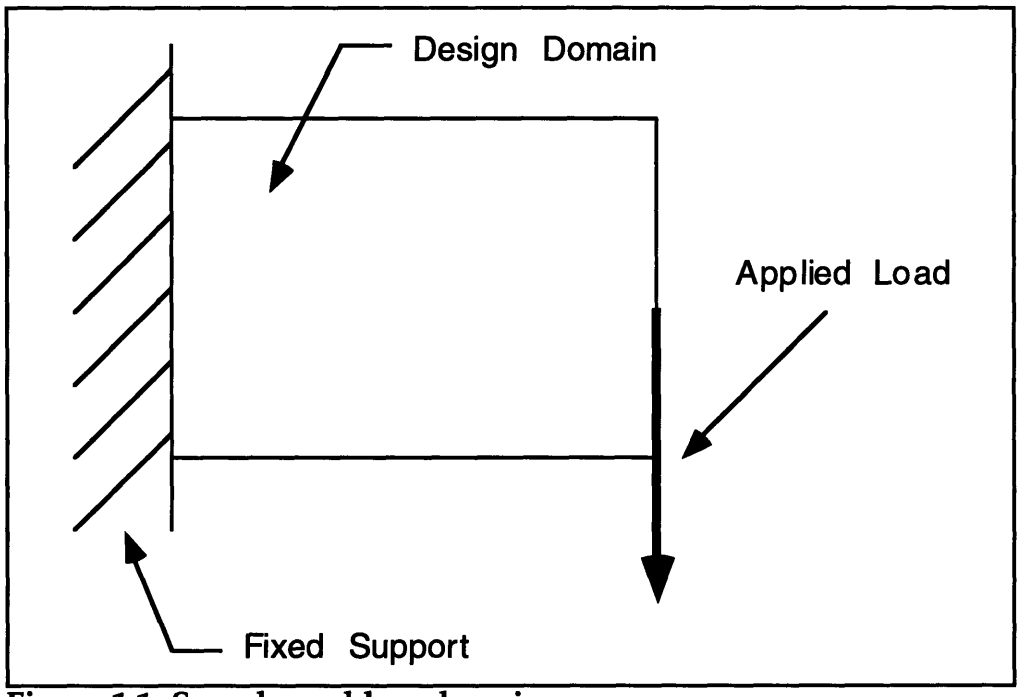


Figure 1.1 Sample problem domain

The goal of the optimization process is therefore to find the optimal distribution of material in a fixed region given certain measures of design performance and a set of design and/or performance constraints. In this study, the problem definition is expanded, in that it is desired that the optimization process produce a diverse set of nearly-optimal designs instead of a single, "best" design. The optimization technique, a genetic algorithm, uses an evolutionary process based on the concept of "survival of the fittest" to take a population of several possible designs and "evolve" it into a set of optimal or nearly-optimal structural topologies. In this investigation, this evolutionary optimization model is extended by making provisions that cause "species" to emerge in the population as it evolves. In the biological context, these species are exploiting "niches" in an environment. In the structural context, it will be seen that these species correspond to local optima that are distributed across the search space. Importantly, a speciated GA search does not need to be significantly more computationally expensive than one that does not reveal species. In fact, this study develops a number of methods for improving GA efficiency and thereby

reducing run-times. Additionally, a new optimization technique which combines genetic algorithms and simulated annealing to optimize the design of adaptive structures is introduced.

1.5 Organization

This thesis presents the development and application of an improved genetic algorithm-based system for structural topology optimization. It is divided into chapters as follows:

Chapter 2, Genetic Algorithm Basics, provides an introduction to the workings of genetic algorithms, including design representation and optimization methods.

Chapter 3, Extensions to the GA, describes modifications to the basic GA technique and the effects of altering various GA parameters. Genetic algorithm improvements include overlapping populations, non-standard chromosomes, and fitness sharing.

Chapter 4, Structural Optimization, provides a basic introduction to structural optimization problems, including size, shape, and topology optimization.

Chapter 5, Related Work, gives more specific details on previous work in structural topology optimization, with particular attention to previous efforts using genetic algorithms.

Chapter 6, This Investigation, begins by explaining the differences between this study and previous efforts. The details of this investigation are then presented.

Chapters 7 and 8 detail the application of this approach to sample problems and demonstrates the improvements in optimization performance. Chapter 7 presents the results of the modified genetic algorithm approach applied to the design of planar stress structures. Chapter 8 demonstrates a hybrid genetic algorithm / simulated annealing based approach for the design of adaptive structures.

Chapter 9, Conclusions, reviews the contributions of this work, discusses the results presented in chapters 7 and 8, and offers ideas for future work in this area.

Chapter 2

Genetic Algorithm Basics

2.1 Overview

The genetic algorithm is the principal optimization method used in this study. This chapter introduces and describes the fundamentals of the genetic algorithm, including common design representations which may be used with genetic algorithms, design evaluation and selection methods, and the basic operators needed to implement a genetic algorithm.

2.2 Genetic Algorithm Fundamentals

2.2.1 Introduction

Genetic algorithms (GA's) are search algorithms modeled after biological systems in which a population of organisms evolves through the process of natural selection (Holland, 1975). In a genetic algorithm, an organism is typically represented by a set of data referred to as a chromosome. For a design optimization problem, this data would usually contain all of the information needed to define one possible design (one organism is analogous to one design, or one point in the search space). The GA assigns a score to each chromosome in the population based on how well the design it represents meets the design requirements. These fitness scores are used to probabilistically select chromosomes to be "parents" for the next generation of individuals, modeling the natural phenomenon of "survival of the fittest." Various probabilistic operations modeled after genetic crossover and mutation are performed on these "parent" chromosomes to produce a new population of possible designs. Optimization occurs as several "generations" of this process of evaluation, selection and reproduction improve the quality of the designs. Typically, as the number of generations increases, the fitness of the population (which can be measured in a number of ways) increases and the diversity of the chromosomes within the population decreases as the GA "converges" to one optimal or nearly optimal solution.

One advantage of genetic algorithms is that they represent a compromise between "weak" and "strong" search methods. "Strong" search methods, such as numerical optimization methods, use auxiliary information such as function gradients to perform search in an informed manner. "Weak" methods, such as random or exhaustive searches, sample the design space extensively, but in an uninformed manner. Strong methods are more efficient, but are also more likely to settle at a local optimum, and they often require design space continuity and derivative existence. Weak methods are more robust and less demanding with regards to the search space, but they are much less efficient. Genetic algorithms, in contrast, operate with a "strong" progression towards improved designs, but use the "weak" operators of probabilistic selection, crossover and mutation to search for a global optimum.

2.2.2 Design Representation: Designs Encoded as Chromosomes

When using genetic algorithms in engineering design problems, the characteristics of a possible design must be encoded into some type of chromosome (see, *e.g.*, Goldberg, 1989a). Genetic algorithms are very flexible in this respect, as many different schemes are possible to perform the mapping from genotype (chromosome) to phenotype (design).

The most common method for encoding a design into a chromosome is to represent the design by a fixed number of parameters then convert each of these parameters into a string of binary digits (bits). Concatenating all of these bit strings in a fixed order forms the chromosome. Each bit position can be thought of as a gene, the value of the bit can be thought of as that gene's allele. For example, suppose a GA was to be used to optimize a set of three design parameters. As shown in figure 2.1, a chromosome could be created as a set of three binary strings, one for each parameter. The number of bits in each string controls the resolution and/or range of each parameter, allowing the parameters to be integers (parameters 1 and 3) or real numbers (parameter 2) as desired. A simple binary-to-decimal conversion function is all that is needed to go from chromosome to design. For example, mapping a 5-bit binary number (such as parameter 2, Y, in figure 2.1) to a real number in a given range is achieved as follows. First, map the binary number to its integer equivalent (01100 = 12) and then divide by the maximum value of the binary number, in this case, $(2^5 - 1) = 31$. Finally, multiply this result by the size of the parameter range (1.0 in the case shown in figure 2.1). So, in the case shown in figure 2.1,

$$Y = \frac{12}{31} * 1.0 = 0.39 \qquad (2.1)$$

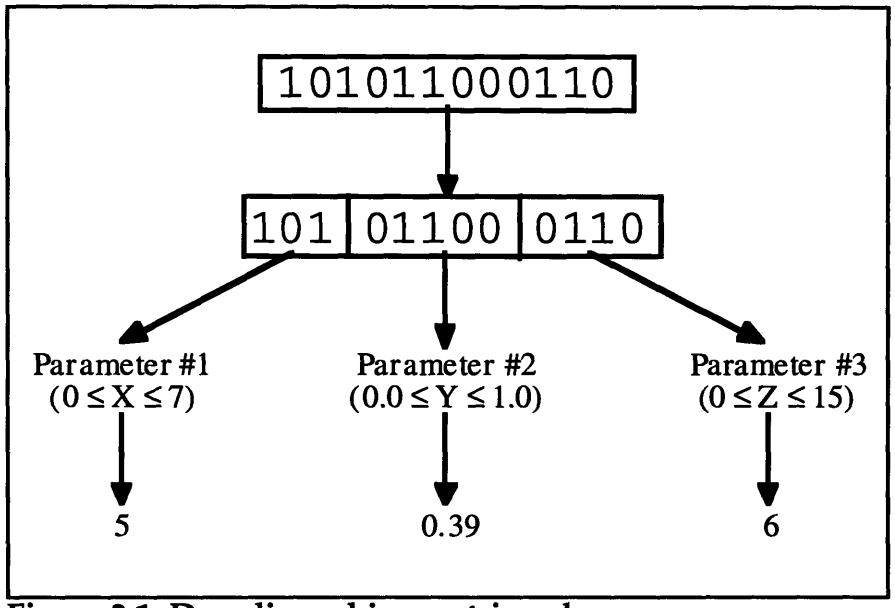


Figure 2.1 Decoding a binary string chromosome

Several other types of chromosomes may be used to represent a design. Ordered lists, strings of characters, multi-dimensional arrays, and tree-shaped structures have all been successfully implemented as types of chromosomes for genetic algorithm search.

It is also important to note that the size of the chromosome (*i.e.*, the amount of information encoded into it) controls the size of the search space. For example, in the case described above, the number of bits used in the binary string chromosome determines the size of the search space. The number of possible values of an n-bit binary string chromosome is 2^n. As a result, increasing the resolution and/or range of parameter values can increase the size of the search space dramatically, leading to significantly longer run times and more required computations.

2.2.3 Optimization: Evaluation, Selection and Reproduction

A genetic algorithm typically begins by randomly creating an initial population of chromosomes (potential designs). After that, most of the work done by the GA consists of three operations: evaluating chromosomes, selecting chromosomes to reproduce, and performing various reproduction operations on the selected chromosomes to create new individuals.

Evaluation

In order for a GA to implement the idea of "survival of the fittest," a function must be created which can assign a score to each chromosome based on its "fitness." This fitness function can generally be thought of as a measure of the quality that is to be maximized. Of course, a GA can also be used to perform a minimization. In this case, the fitness function must be set up so that fitness rises as the quantity to be minimized (often the objective function) decreases. This is typically done by setting the fitness value equal to the reciprocal of the objective value, or equal to some large constant minus the objective score.

Typically, a fitness function evaluates the chromosomes one at a time, using some absolute measure of performance. For instance, the stiffness-to-weight ratio of each structure might be evaluated. In other cases, when an absolute scale is difficult to define, the fitness function may evaluate the chromosomes relative to one another instead.

It is common in the early generations of a GA run for a few individuals to have vastly superior fitness scores. This can cause the GA to prematurely converge to a local optimum. As a GA runs longer, the population's average fitness tends to come closer and closer to that of the best individual(s). In this case, it becomes more and more difficult for the GA to pick out the "fittest" individuals, and so further improvement becomes difficult. To help eliminate these problems, some type of fitness scaling is nearly always used (Goldberg, 1989a, pgs. 122-124). This scaling helps to keep fitness values closer together during the early generations, then spread them further apart as the run continues. Several schemes exist to perform this scaling. The most common ones include linear scaling, sigma truncation scaling, and power law scaling. Other scaling rules, such as fitness sharing, have been created to help encourage population diversity. This topic will be discussed further in section 3.3.3.

Selection

Once the initial population has been created and evaluated, individuals must be selected for reproduction. In order to improve the overall fitness level of the population, chromosomes are selected based on their fitness scores. Chromosomes with high fitness scores are more likely to survive and reproduce, those with low scores are more likely to "die off."

Selection is generally biased towards those chromosomes with high fitness scores. The simplest and most common way of implementing this is by using "Stochastic Sampling

with Replacement" (Baker, 1987), more commonly called "roulette wheel selection." With this algorithm, a "roulette wheel" is created with a number of "slices" equal to N, the number of chromosomes in a population. The size of each chromosome's "slice" is given by:

$$size(a) = \frac{fitness_a}{\sum_{i=1}^{N} fitness_i} \qquad (2.2)$$

This wheel is then randomly "spun" N times, with each spin selecting one chromosome to serve as a parent for the new population. It is important to note that a given chromosome can be selected more than one time per generation. As a result, even though N parents are chosen from a population of N chromosomes, not every chromosome will be selected, and there is no guarantee that any particular chromosome will be selected. The possibility that the best chromosome will not be selected can be avoided by using an "elitist" scheme. This will be discussed further in the following section.

Reproduction

Once all of the parent chromosomes have been selected, they must "reproduce" to create a new generation of children. This chromosome reproduction is modeled after genetic reproduction and usually consists of two operations: crossover and mutation.

Crossover involves combining different parts of two parent chromosomes to make two new "children." The simplest and most common crossover operation is single-point crossover, as shown in figure 2.2. First, a crossover site is randomly selected along the length of the chromosomes. Then, the front of one chromosome is appended to the back of the other chromosome and vice versa. Thus, two children are created from two parents. Generally, most but not all parent chromosomes undergo crossover. Those few parents that do not crossover enter the new population unaltered. In "elitist" GA's, the best chromosome generated so far will always proceed unaltered into the next generation.

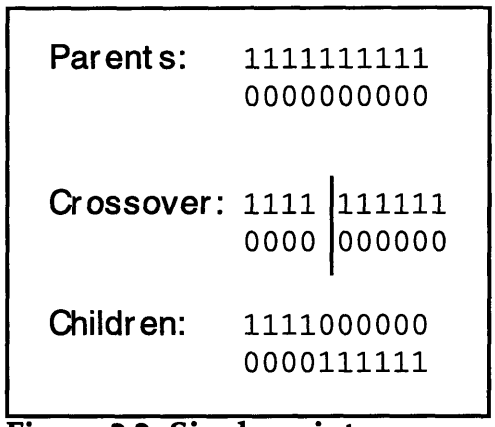


Figure 2.2 Single-point crossover

Crossover operators can create new and better children, but they do not have the ability to create any "new" genetic material. All of the children are made up of parts of their parents. As in nature, mutation is used to introduce random new genetic material into the population. Mutation is performed with a very low probability on the children. One bit per thousand, for instance, might be randomly chosen and inverted or otherwise altered.

2.3 Summary

The genetic algorithm model described above can be thought of as the basic or canonical approach. This approach is a robust and general search and optimization technique that can be applied to a variety of problem domains. Because a GA is a zero'th order method, requiring no derivative or gradient calculations, it is well-suited for discrete and/or multi-modal design spaces. There are, however, some drawbacks to such an approach. Although the parallelism inherent in a GA helps it to avoid local optima, a GA as described above does tend to converge to a single solution. Also, a genetic algorithm usually requires that a large number of designs be evaluated, often resulting in large computation times. As a result of these short-comings, some extensions to the basic approach have been created, as described in the following chapter.

Chapter 3

Extensions to the GA

3.1 Overview

This chapter presents extensions to the basic GA approach discussed in the previous chapter, including different design representations and genetic algorithm operators. Particular attention is paid to genetic algorithm modifications aimed at maintaining a diverse population of solutions. Finally, the effects of various genetic algorithm parameters are discussed.

3.2 Modifications to the Basic Approach

3.2.1 Introduction

While the approach described in the previous chapter is a robust and general method applicable to many search and optimization problems, many other problems require some modifications for improved performance and efficiency. Genetic algorithms with overlapping populations may be used to speed up the optimization process in cases where evaluating a design is computationally expensive. More complex design representations are necessary in some cases to provide a more effective design-to-chromosome mapping. Various other GA functions and parameters may also be modified to improve performance in certain cases. In fact, it is safe to say that alternative algorithms and mechanisms have been proposed and implemented for nearly every aspect of the basic GA approach described previously.

3.2.2 Overlapping Populations

One of the more commonly used variants of the simple GA is the "steady-state" GA (Syswerda, 1989, 1991). The difference is as follows. A simple GA uses non-overlapping populations. Each generation, a whole new population is created and the old population is discarded. A steady-state GA, on the other hand, uses overlapping populations. A smaller

number of parents are chosen each generation, and the children they create are inserted into the original population, replacing some of the individuals there. Typically, the individuals with the lowest fitness scores are replaced, although other schemes have been used. Using overlapping populations requires fewer fitness computations per generation and correspondingly more generations overall, but typically results in run-times several times faster than a simple GA with non-overlapping populations.

3.2.3 Non-standard Chromosomes

Although the binary string chromosome described in section 2.2.2 is used for a variety of problems, several other types of representations have been used and found to work well. For example, the alleles in a string need not be binary bits. Alleles could be integers, characters, or real numbers. In some cases, the alleles in a string are labels representing a task where the order of the labels indicates the task sequence (see Davis, 1995). The chromosomes do not even have to be strings. In genetic programming (see Koza, 1992), tree structures are used to represent the decision structure of a computer program. These structures are decoded to reveal their phenotype: computer code that can be run to determine its fitness. The result is that software can be "evolved" by a GA. Another variation on the binary string chromosome is the multi-dimensional array chromosome. In its simplest form this is just an array of binary bits, but again, these alleles could be integers, real numbers, *etc.*

When an alternative representation is used, new crossover and mutation operators sometimes are required. In this study, two-dimensional array chromosomes are often used. Single point crossover needed to be redefined, as shown in figure 3.1a. A single point is randomly selected somewhere along the height and width of the chromosome. The two parent chromosomes are divided into four pieces each, and the complementary pieces are exchanged to create the two children. Other crossover operators, such as two-point crossover (see figure 3.1b), have been defined for two-dimensional array chromosomes, but in this study, single point crossover was found to be the most effective, and was used throughout.

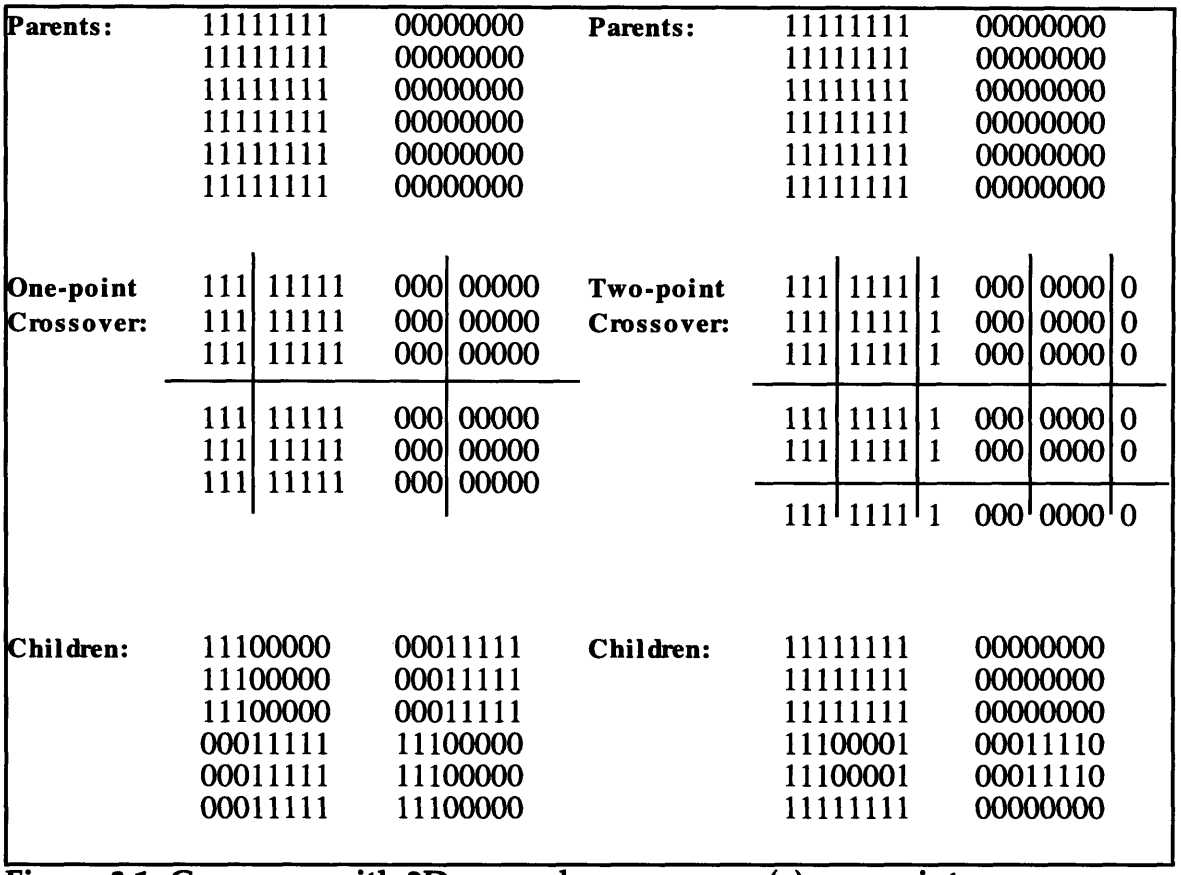


Figure 3.1 Crossover with 2D array chromosomes: (a) one-point crossover and (b) two-point crossover

3.3 Fitness Sharing and Speciation

3.3.1 Introduction

When a GA is used in a problem with many equally good solutions, it is usually desired that the GA find a number of these solutions. However, a standard GA used for such a problem will have a final population which will be located predominately around just one of these optimal solutions. This degraded performance is due to the finite size of the population and the accumulation of stochastic errors and has been given the name "genetic drift" (Goldberg, 1987). Clearly, it is desirable that the GA not converge to just one of several equally good designs. In fact, it is often desirable that the GA find any near-optimal designs in addition to the global optimum. Genetic algorithm fitness functions are not always exact measures of quality, and a design which has a slightly lower fitness score than the "optimum" may actually be more desirable due to factors which were difficult to model in a fitness function. In cases such as this, it is desired that the GA find a number of

solutions, all nearly optimal according to the fitness function. The designer may then select the design or designs that are the best.

3.3.2 Theory of Fitness Sharing

One method for improving GA performance in multi-modal domains is the use of fitness sharing (Goldberg and Richardson, 1987). Fitness sharing is based on an analogy to biological systems. In nature, different species evolve to exploit different niches in their environment. Similarly, with fitness sharing, chromosomes are allowed to evolve into separate clusters, or species, to exploit some local area, or "niche", of the design space. The use of fitness sharing in a GA helps reduce competition between very dissimilar chromosomes, and helps to improve the overall diversity of the population by delaying convergence.

The first use of niche and speciation in artificial genetic search is credited to Holland (1975), who used a modification of the two-armed bandit problem to illustrate his approach. Consider a slot machine with two "arms," each with its own payoff chute. The two arms pay out different amounts, but each pays out the same amount every time it is pulled. Playing this machine is a group of agents, each seeking to maximize individual profit. In the standard, or unshared, case, each agent will try each arm once, determine which one pays more, and position itself at that arm for future trials. This is just a simple optimization problem with only two points in the design space. If the agents reproduce according to fitness, more and more agents will "line up" in front of the arm with the better payoff, and the population will converge to that location.

This last example showed no signs of niche or species-like behavior. To induce such behavior, we introduce the notion of fitness sharing. In this shared case, each agent must share its payoff with all of the other agents at the same arm (all other agents in the same niche). As a result of this change, the population of agents will tend to distribute itself so as to maximize individual payoffs. For example, if their are 100 agents, and the two arms pay out $75 and $25, respectively, the population should settle to a state in which there are 75 agents at the first arm, and 25 at the second. Each agent will have a payoff of $1, and no agent could improve its payoff by changing arms.

Extending this problem to the "k-armed bandit" does not change the results. In a system with k points in the design space (k arms and k lines), equilibrium will be reached when the

ratios of payoff to line length are all equal (Holland, 1975). The use of fitness sharing causes the formation of stable subpopulations (species) at various areas in the design space (niches), where the size of each subpopulation is proportional to the fitness of its niche. In a typical GA problem, each agent is represented by a chromosome, and the payoff of an arm is analogous to the fitness of a design. Some difficulties arise, however, in determining which niche a chromosome is in and how its fitness should be shared.

3.3.3 Speciating Genetic Algorithms

A speciating genetic algorithm can be defined as any GA in which efforts are made to maintain diversity within the population. The primary unique feature of most speciating genetic algorithms is the use of a sharing function to perform fitness scaling. The basic role of the sharing function is to define what a "niche" in the design space is and to scale the fitness scores of the individuals within a niche.

In order to use a sharing function, some measurement of distance (dissimilarity) between chromosomes must be defined. Generally, this distance function takes the form:

$$d_{ij} = d(x_i, x_j) \tag{3.1}$$

where d_{ij} is the distance between chromosomes i and j. x_i and x_j can either be the genotypes or phenotypes of chromosomes i and j. Better results are usually obtained by comparing phenotypes (it's better to compare designs than to compare the encodings of these designs). Typically, this distance function is normalized, so that two identical chromosomes have a distance of 0, and the two most different chromosomes possible have d_{ij} equal to 1.

Once a distance function has been defined, a sharing function is defined to determine the extent to which two chromosomes share their fitness. A sharing function, s(d), is also normalized between 0 (for the most dissimilar chromosomes), to 1 (for two identical chromosomes). For example, the fitness of a given chromosome, i, could be given by:

$$fitness_{shared}(x_i) = \frac{fitness_{raw}(x_i)}{\sum_{j=1}^{n} s(d(x_i, x_j))} \tag{3.2}$$

Note that because a chromosome always shares its fitness with itself and $s(d(x_{i,i}))=1$, the denominator can never be less than 1. A fitness function such as this one will cause

evolving chromosomes to be concentrated around different peaks in the design space in numbers proportional to the relative fitness values of the peaks.

Goldberg and Richardson (1987) proposed using power law functions to determine the amount of sharing between two chromosomes:

$$s(d) = 1 - (d/\sigma)^{\alpha}: \qquad d < \sigma \tag{3.3}$$

$$s(d) = 0: \qquad d \geq \sigma \tag{3.4}$$

Plots of such functions are shown in figure 3.2. In these equations, σ and α are constants. The constant σ determines the size of a niche, α determines the shape of the curve. Chromosomes only share their fitness with chromosomes less than σ away.

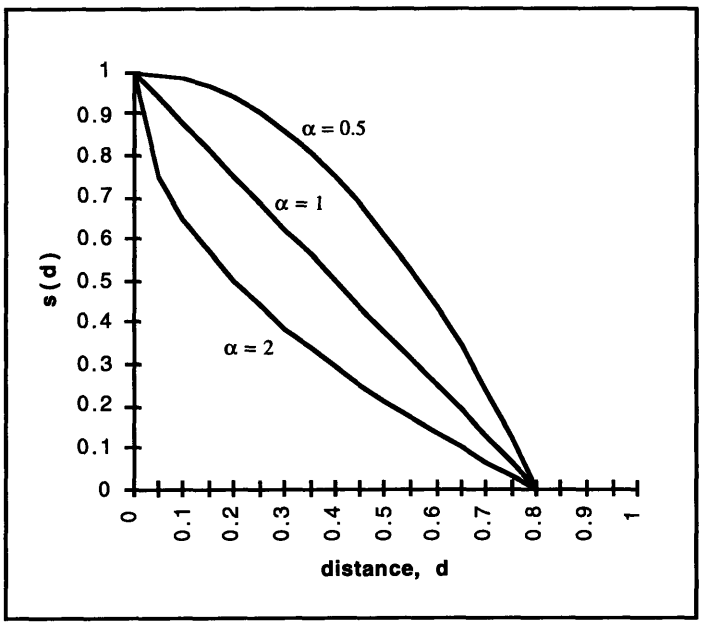


Figure 3.2 Power law sharing functions, $\sigma = 0.8$

In this case, discrete niches are not defined. Instead, chromosomes share their fitness with whatever chromosomes are "nearby." This method has been found to work particularly well for problems with continuous search spaces. However, in problems with discrete design spaces it can become difficult to decide where to draw the line between separate subpopulations.

3.3.4 Other Niching Methods

Methods other than fitness sharing have been proposed for maintaining a diverse population of solutions. One of the earliest attempts at avoiding premature convergence was DeJong's crowding mechanism (DeJong, 1975). This technique uses overlapping populations, where a new population member (a child chromosome) tends to replace the most similar chromosome in the previous population. This technique was found to slow convergence of the GA to a single solution, but also to limit the exploration of new areas of the design space. Subsequent studies have shown that crowding is not a very effective method for discovering multiple local optima (Deb & Goldberg, 1989; Mahfoud, 1995).

Another approach is to restrict competition among dissimilar chromosomes during the selection process. Mahfoud (1995) describes such a method, using a form of tournament selection. This process proceeds as follows. First, two similar chromosomes are randomly selected from the population. (One is randomly selected, then potential competitors are randomly selected until one is within the specified distance of the first). The fitness scores of the two competitors are compared, and the chromosome with the higher score is used as a parent for the next generation. Several variations of both of these schemes, such as deterministic crowding, have been implemented, although fitness sharing remains the most widely used technique for finding multiple optima. (See Mahfoud, 1995 for a thorough review of other niching methods for GA's.)

3.4 Optimization Parameters

Several methods and parameters must be chosen in order to implement a genetic algorithm. Some of the most important GA methods and parameters are the following:

GA Type:	Simple or Steady-State (overlapping or non-overlapping populations)
Population Size:	Number of chromosomes in a population
Chromosome Type:	Binary String, Array of real numbers, ordered list of integers, *etc.*
Fitness Scaling Method:	Linear Scaling, Power Law Scaling, Fitness Sharing, *etc.*
Selection Method:	How parents are chosen from the population

Crossover Operator:	How two chromosomes are combined to produce children
Mutation Operator:	How a gene is mutated
Percent Crossover:	Percent chance that a particular parent will undergo crossover
Percent Mutation:	Percent chance that a particular gene will undergo mutation

The methods and parameter values selected can have a great impact on the effectiveness and the efficiency of the algorithm. Unfortunately, there is no clearly-defined universal process for performing this selection. Several studies (see, for example, Schaffer *et al.*, 1989, or Goldberg, 1989b) have been done to investigate the effects of different parameters on GA performance, and, in some cases, the results of these studies can be used to find an "optimal" set parameter values. However, the "optimal" parameters change from problem to problem, and the selection of parameter values for a GA is often an intuitive and empirical process with many tradeoffs to consider. For example, having a small population size will result in a GA with fewer function evaluations and a faster convergence rate, but the population may not contain enough genetic information to effectively explore the search space. Using a larger population may be necessary to explore a larger portion of the search space, but this will result in longer run-times. Similarly, high rates of crossover and mutation help to explore a greater amount of the search space and encourage diversity. However, too much mutation can degrade GA performance, making it act like a random search. Choosing GA methods is generally somewhat easier and more intuitive, based on what sort of results are desired. For example, fitness sharing is more likely to produce a diverse set of solutions, whereas a linear or power-law fitness scaling method will be more efficient at finding a single "best" solution. Further discussion of parameter selection will be discussed with regards to specific problems in chapter 6.

3.5 Summary

Many modifications can be made to the basic genetic algorithm approach in order to improve performance on a particular design problem. Overlapping populations can be used to decrease the number of fitness evaluations and run-times. Different types of chromosomes can be used to encode designs more effectively and improve GA performance. If a diverse set of nearly-optimal solutions is desired, fitness sharing can be implemented. Many alternative methods have been proposed for nearly every aspect of the

genetic algorithm (selection, crossover, *etc.*). Choosing which methods to use and setting the values of various GA parameters can have a great impact on performance and efficiency. Selection of most of these methods and parameter values is very problem-dependent, and no universal rules have been defined for optimal parameter selection. However, with an understanding of how GA's work and a little bit of experimentation, changing these parameters and methods can result in tremendous improvements in the performance of a genetic algorithm.

Chapter 4

Structural Optimization

4.1 Introduction

Structural optimization is the design of a mechanical component so as to maximize the component's utility, where utility is based on functional requirements and/or performance constraints that are structural in nature. In general, the component's design must be represented by some finite number of design variables. The utility, or fitness, of a design is evaluated using only these variables. This fitness value is a measure of how good the structure is and can include factors such as weight, performance data, size, manufacturability, reliability, *etc*. Constraints may also be applied to the problem, placing limits on what structures are acceptable. For example, the designer could place limits on the maximum allowable stress within a structure or the maximum size of a component. In general, the goal of structural optimization is to design a mechanical structure that best meets the functional objectives while also satisfying the given constraints.

4.2 Problem Formulation

In order to perform structural optimization, the qualitative problem description (*e.g.*, minimize weight of a beam subject to a known load and displacement constraint) must be converted into a quantitative mathematical statement (Arora, 1989). The first step in this conversion is the definition of the design variables, the parameters that determine the design of the structure. Each design variable represents one quality of the component's design which can be varied in order to find the optimum structure. There are two basic types of design variables: those related to material properties and those related to the structures geometric and shape properties. Material properties could include factors such as density or yield strength, shape variables determine what the structure looks like. Mathematically, it is important to note that these design variables can be either continuous or discrete. For example, suppose that part of the structure was to be made of a beam with a circular cross-section. The diameter of this beam might be a discrete variable, since beams could only be purchased with certain cross-sections. However, the length of the beam could be represented by a continuous variable, since a purchased beam could easily be cut to

whatever length is desired. Once all of the design variables have been defined, they are combined to form a vector, x, as shown below:

$$x^T = [x_1\ x_2\ x_3\ \ldots\ x_n] \tag{4.1}$$

The n-dimensional space defined by x is known as the design space or search space. Each combination of design variables represents one design, one point in the search space.

Once this vector x has been defined, an objective function must be developed. This function measures the utility, or "goodness," of a given design. Given a set of design variables, the objective function analyzes this design and returns a value indicating its utility. This utility function can be expressed as:

$$U = U(x) \tag{4.2}$$

Depending on the optimization technique being used, the utility function may either be maximized or minimized. In this study, it is always desired that the utility function be maximized.

A set of constraints can also be defined. These constraints are used to determine whether a given design is feasible or infeasible. These constraints can include both design constraints and performance constraints. Design constraints act directly on the design variables, performance constraints require evaluation of the design. An example of a design constraint would be a limit placed on the value of a particular design variable. For example, the length of a beam could be required to be less than three feet ($x_2 < 3$). An example of a performance constraint could be a maximum allowable stress at any point in the structure. Performance and design constraints may be equality constraints ($x_1 + x_2 * x_3 = 1.0$), inequality constraints ($x_1/x_3 > 4.0$), or side constraints ($1 < x_1 < 6$). Thus, the generalized structural optimization problem can be expressed as follows:

Find the vector of design variables, $x^T = [x_1\ x_2\ x_3\ \ldots\ x_n]$

to maximize $U(x)$

subject to p equality constraints,
$f_i(x) = F_i \qquad i=1 \text{ to } p$

m inequality constraints,
$g_j(x) < G_j \qquad j=1 \text{ to } m$

and n side constraints
$A_k < h_k(x) < B_k$ k=1 to n

4.3 Structural Optimization Classifications

Structural optimization problems can be classified by the way in which the design variables represent the design of the structure. Three main types of representations are possible: size, shape, and topology. Depending on the flexibility of the optimization method used, it may or may not be possible to use a mixture of these classes in a single problem.

4.3.1 Sizing Optimization

The simplest of the three types of structural optimization is sizing optimization. In this case, the structure's general shape and topology are held constant, while the specific dimensions of the design are modified. Design variables are used to represent different dimensions of the structure, with the optimal design being the one with the set of dimensions that yields the best performance. An example of sizing optimization would be the design of a truss, in which each member is made of a length of pipe, and each section of pipe is to have the same cross-section. In order for this to be a sizing optimization problem, the layout of the truss must be known in advance. The only design variables are those representing the dimensions of the cross-sections of the pipe. In this case, only two design variables would be needed to represent the pipe dimensions, D_{in} and D_{out}. Note that, in general, other design variables could be used to represent material properties of the pipe. In this case, however, we are assuming that the material properties are known and fixed. The objective function could be to minimize the weight of the truss, subject to a constraint on the maximum allowable stress in the pipe (an inequality constraint) and side constraints on the size of the pipe, $D_{out} < D_{max}$, $D_{in} < D_{out}$. Given this objective function, the structural optimization process would find the combination of dimensions giving the lightest truss capable of meeting the given constraints.

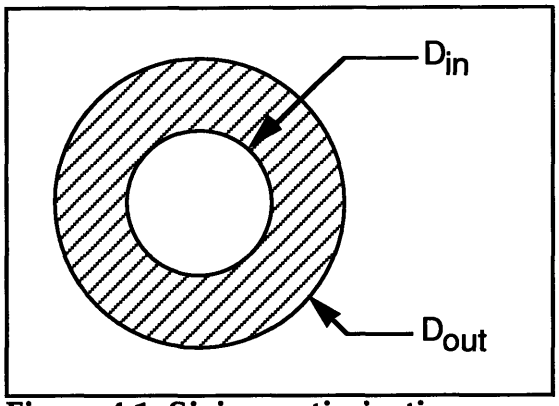


Figure 4.1 Sizing optimization

4.3.2 Shape Optimization

Shape, or geometrical, optimization is a somewhat more complex process. In this case, the design variables generally represent some type of nodal co-ordinates. An example of this would be the optimal design of a truss structure with a given number of nodes and beam elements. In this case, the cross-section of the truss members is known and the layout of the truss is to be optimized. In the two-dimensional case, the design variables would be the x and y co-ordinates of each node. Another example of shape optimization would be the optimization of the cross-section of a member using a collection of B-spline or Bezier curves to represent the cross-sectional shape (Farin, 1993). In the case shown in figure 4.2, the cross-section can be varied, but it can never contain any holes. In general, shape optimization can lead to better results than sizing optimization, but also results in a larger search space and correspondingly longer run-times.

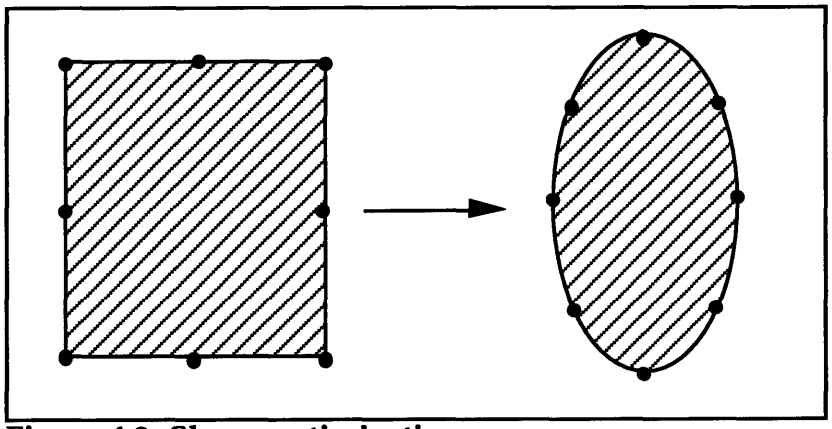


Figure 4.2 Shape optimization

4.3.3 Topology Optimization

The most difficult of the three classifications of structural optimization is topology optimization. In this case, the design variables control the topology of the design. This is also the most general optimization method, as the size and shape of the structure are affected by the topology. The difficulty of implementing this method comes from its generality. Representing the topology of a structure is difficult and generally requires a large number of design variables. A commonly used representation technique for topology optimization is to treat the problem as a configuration design problem, where the overall design is treated as an assembly of a large number of "building blocks." The process begins with a set of all possible building blocks being present, defining the maximum size and shape of the structure, known as the design domain. As the optimization process continues, various "blocks" are allowed to disappear an/or reappear, altering the topology of the structure. Each building block is represented by a design variable. A design variable value of 1 means that the corresponding block is present, a value of 0 means that it is not. With some optimization methods, the design variables can take on intermediate values between 0 and 1, signifying that material of a lower density is present in the corresponding block. Figure 4.3 shows an example of a design domain discretized into a large number of blocks, and one possible result of an optimization process.

As an example of topology optimization, we will again consider the optimization of the cross section of a beam. In this, the most general case, a much larger number of different designs are possible than with other methods, and all designs produced by sizing or shape optimization can also be found using topology optimization, to the level of discretization. The primary disadvantages of this method are that it is very difficult to deal with analytically, and that, in some cases, the design space can be much larger than for the other two types of structural optimization methods. As the design domain is more finely discretized, the size of the search space increases dramatically.

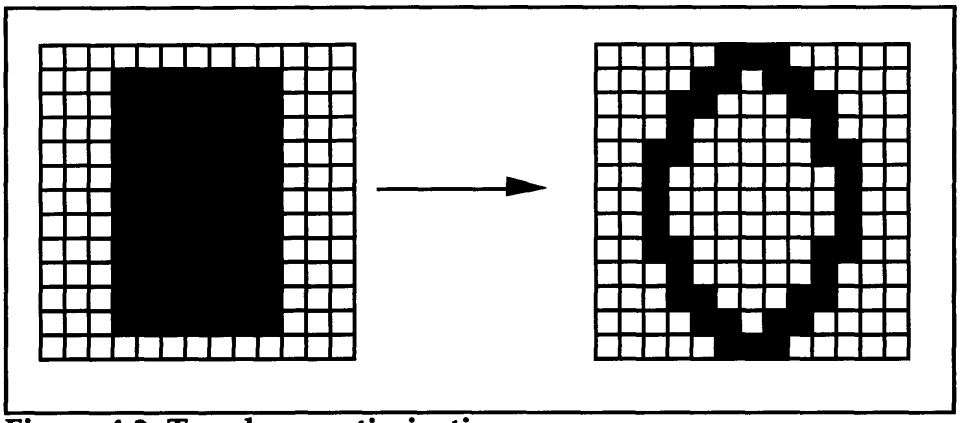


Figure 4.3 Topology optimization

4.4 Structural Optimization Techniques

Once the optimization problem has been defined, a method is needed to perform the optimization. There are two types of methods for carrying out this search, analytical and numerical. Analytical techniques, such as those derived by Mitchell (Mitchell, 1904), typically use calculus and/or variational techniques to find optimal designs. Systems of equations representing the conditions for optimality are found and solved analytically. More recent applications of analytical optimization can be found in Cox (1965) and Hemp (1973). Analytical optimization techniques have been largely theoretical in nature, and have served as a foundation for the more commonly-used numerical methods (Lev, 1981).

Numerical techniques have been developed to deal with more complex problems and constraints. These methods numerically model the design of a structure then repeatedly analyze (usually using a finite element analysis) and alter this model until some near-optimal design is found. Such methods generally can not ensure that a global optimum has been found, but they are much more versatile and easily applied than analytical methods. Several common types of numerical optimization have been used for structural optimization, including mathematical programming (see Arora, 1989 or Kirsch, 1993), fully stressed design (see Gallagher, 1973), and the optimality criteria method (see Berke and Venkayya, 1974). More recently, some other techniques have been applied, such as homogenization methods, simulated annealing, and genetic algorithms. These methods will be detailed further in the following chapter.

4.5 Design of Adaptive Structures

Recently, innovations in the development of actuator technologies have led to an increasing interest in the design of adaptive structures, *i.e.* structures whose properties change in response to changes in operating conditions. Such structures are equipped with some combination of sensor and actuator, such as a piezoelectric ceramic material, a heat-activated shape memory alloy, or a magnetostrictive material, which can detect a change in conditions and then somehow alter the structure in response to this change. For example, suppose a piezoelectric material was embedded in or affixed to a structure. Applying a load to the structure could induce a strain in the piezoelectric material, causing it to produce a current. This current could be used to drive a logic circuit controlling other actuators within the structure. A commercial example of this is the so-called "smart ski" developed by K2. This ski uses embedded piezoelectric actuators to first detect then dampen shocks and vibrations, helping to keep the ski in contact with the snow (Ashley, 1995).

4.6 Summary

The problem of structural optimization involves designing a mechanical structure so as to maximize the value of the structure's utility while ensuring that the structure can meet certain constraints on its performance and geometry. This is accomplished by defining a number of design variables, which fully represent the parameters of the structure's design, then defining an objective function to determine the goodness of a design based on the values of these design variables. The design of a structure can be represented in three different ways, as a set of dimensions, a set of control points, or as a discretized domain. Increasing the number of design variables will generally increase the size of the search space and result in a more "global" optimum solution, but can also result in very long run-times, regardless of the method used to perform the optimization. These problems may be solved either analytically or numerically. Analytic techniques can ensure that the global optimum is found, but these techniques are often not applicable to more complex problems.

Chapter 5

Related Work

5.1 Introduction

Structural optimization has been an active area of research for some time (Haftka and Granhdi, 1986). In more recent years, with the increasing availability of computational power, the specific problem of topology optimization has received a growing amount of attention. A number of different approaches to this problem have been taken, including both continuous (homogenization-based methods) and discrete (simulated annealing and genetic algorithms) methods. This chapter presents an overview of the basic techniques for structural optimization currently being researched.

5.2 Homogenization-based Methods

The most widely-used continuous variable approach to structural optimization problems is the variable density approach, based on material homogenization methods (Strang and Kohn, 1986). Bendsøe and Kikuchi (1988) and associated researchers have applied these techniques to a variety of structural topology optimization problems, with a good deal of success (see, for instance, Suzuki and Kikuchi, 1991 or Bendsøe and Kikuchi, 1988). This method involves discretizing the design domain into a large number of elements, where each element contains a number of microvoids of a particular shape. The design variables for this technique are the size and orientation of the microvoids in each element. These values are allowed to vary continuously and are used to control the density and structural properties of the material within the element. Loads are applied to the structure, and its compliance is measured using a finite element analysis. Typically, this technique attempts to minimize the compliance of a structure given a specified mass of material that may be present in the design domain (*i.e.*, find the optimal distribution of a fixed quantity of mass). An iterative non-linear optimality criteria technique is used to determine the set of microvoid sizes and orientations that will minimize the structure's compliance.

Strang and Kohn (1986) recommend the use of composites in such structural optimization problems, stating that a material/void, or 0-1, dichotomy results in an ill-posed minimization problem, where the optimal solution is difficult if not impossible to obtain. They suggest that a "relaxation" of the problem, modeling the material in the design domain as a composite with continuously variable density, is necessary to transform the original problem into one that can be solved by common optimization techniques.

A number of different models for the microstructure of an element have been proposed for use with material homogenization methods. The microstructural model is used to determine the structural properties of the composite material. All models allow the complete rotation of a microvoid, and allow for the full range of material density from 0 to 1. The simplest of these models is that of a unit cell with a rectangular "pore," or hole, in the center of the cell, representing a void area (Bendsøe and Kikuchi, 1988). Each element in the design domain is made up of several of these unit cells, the material properties of which depend on the size and orientation of the voids within. More recently, other researchers have studied the use of a "Rank-2" layering, which creates a two-scale orthogonal laminate. The volume fractions of strong and weak material define the composite's density and structural properties.

Optimality criteria methods are typically used to solve the minimization problem created using the above model. This is an iterative process which gradually alters the microstructure of an initial design until the criteria for optimality are met and a single "best" design has been generated. There is no guarantee that the result of this method is a global optimum, and using different micro-structural models or different initial designs can yield different final results (Bendsøe *et al.*, 1993).

Another problem with homogenization-based methods is that elements of various densities may still exist in the "optimal" design, requiring some post-processing operation to eliminate material of intermediate densities. Papalambros and Chirehdast (1990) have used binary image analysis approaches to extract precise topological boundaries. The boundaries of the resulting shape can be modeled using spline curves and optimized using more traditional shape optimization techniques.

5.3 Simulated Annealing

One discrete variable approach to structural topology optimization is the use of simulated annealing. This approach has been used by Anagnostou (1992) for structural topology optimization problems involving strength, manufacturing, and heat transfer considerations.

Simulated annealing is a heuristic global optimization method based on statistical mechanics (Kirkpatrick *et al.*, 1983). In this technique, an initial set of design variables is first brought into a high-energy state, or "melted," by a high "control temperature." At this high energy state, the design variables can change (the design can move about in the search space) easily, as atoms at a high energy state move throughout their domain with ease. Optimization proceeds by gradually lowering the control temperature to a minimum value. As in an actual annealing process, as the temperature is lowered, it is hoped that the design variables will change values so as to produce a final low energy state, corresponding to a globally optimum design.

With this technique, the design domain is initially discretized into many rectangular elements, where each element is binary (it can be either material or void, no intermediate densities are possible). An initial candidate design is generated by assigning values of material or void to each element in the design domain, defining an initial topology as shown in figure 5.1. An energy function is then defined for this domain. The energy function measures how well a given design meets the performance objectives and satisfies any constraints. Because simulated annealing is inherently a minimization process (energy of the system is gradually dropping), the energy function must be set up so that energy decreases as design quality increases.

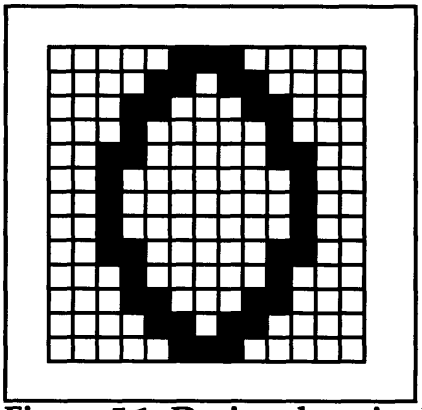


Figure 5.1 Design domain discretized into material/void elements

Once an initial design is defined, its energy is measured, and a new design in the "vicinity" of the current design is produced. This is achieved through the use of a stochastic neighborhood operator, similar to genetic algorithm mutation. Such an operator produces a new design which is similar to the old design, but slightly different. The energy of this new design is then computed. If this energy is lower than that of the previous design, the new design is accepted and replaces it. If the new design has a higher energy, E', it will be accepted only with a probability, P, given by:

$$P = e^{-\frac{E'-E}{T}} \tag{5.1}$$

where T is the current control temperature. After the new design is either accepted or rejected, this process of generating a new design, evaluating it, and either accepting or rejecting it is repeated for a pre-determined number of iterations. Then, the control temperature is lowered, and this entire process is iterated until the temperature reaches its minimum value.

Initially, when the control temperature is high, the algorithm explores new areas of the search space, as the probability of acceptance for a new design is high. As the temperature drops, the probability of acceptance also drops, and the algorithm focuses more on the fine-tuning of an already good design. The rate at which the temperature drops (the number of iterations per temperature) has a large impact on the quality of the solution produced by this method. A rapidly-cooling temperature will require fewer energy evaluations, but will likely produce a less good solution than a slower-cooling attempt.

5.4 Genetic Algorithms

Another recent approach, which is the focus of this study, is the use of genetic algorithms to perform structural optimization. In this case, optimization is performed by "evolving" a population of chromosomes, as described in chapter 2. Genetic algorithms have been used to perform sizing, shape, and topology optimization. This section provides a brief review of GA-based sizing and shape optimization methods and a more thorough review of previous efforts to perform topology optimization using genetic algorithms.

5.4.1 Sizing Optimization

One of the earliest uses of GA's for structural optimization was the sizing optimization of a 10-member plane truss, performed by Goldberg and Samtani (1986). In this study, the weight of the truss was to be minimized, subject to constraints on the allowable levels of stress in the truss members. The geometry and topology of the truss structure were held constant, and the only variables were the cross-sectional areas of the ten members, which were treated as continuous variables. This work was later extended by Rajeev and Krishnamoorthy (1992), who used genetic algorithms to optimize the cross-sectional dimensions of 10-, 25-, and 160-bar truss structures. In this study, the cross-sectional areas of the bars were treated as discrete variables, and so the problem became a type of catalog-based search problem. Further work in GA-based sizing optimization has been performed by Hajela (1990). Again, the cross-sectional areas of a ten-member truss were optimized; this time the truss was subjected to dynamic loading conditions. These examples demonstrate that the genetic algorithm can be an effective optimization tool for problems with a variety of different types of fitness and constraint functions and continuous or discrete variables.

5.4.2 Shape Optimization

Genetic algorithms have also been used to perform shape optimization for structural optimization problems. In typical shape optimization problems, the topology of the structure is fixed and control points are used to define its size and shape. Jenkins (1991a, 1991b) used genetic algorithms to optimize the shape of several discrete-member truss structures. For example, an 18-member truss structure was optimized so as to minimize weight subject to stress constraints. In this example, the design variables were the locations of the truss nodes. In other examples, such as that of Watabe and Okina (1993), the design variables are a set of control points, not the structure's nodes. These control points controlled the node locations using a technique called Free-Form Deformation.

Richards and Sheppard (1992) used a classifier system to perform shape optimization. A classifier system (Goldberg, 1989a, pgs. 217-230) uses rules to govern its behavior. These rules are learned using a genetic algorithm, *i.e.*, *rules* evolve. In their study, Richards and Sheppard optimized the shape of a two-dimensional structural component. Again, the goal was to minimize mass subject to a maximum stress constraint. The design variables were a set of control points. These control points were fit with cubic spline curves to define the

component's outer boundary. The GA was used to learn which control point modifications resulted in a shape of optimal performance.

5.4.3 Topology Optimization

More similar to the work in this study are the following investigations, which attempt to find optimal structural topologies using a genetic algorithm. Like the homogenization and simulated annealing based approaches described earlier in this section, these GA-based methods use a discretized design domain (see figure 5.1 above) and try to optimize the distribution of material in this domain. The earliest work done using GA's for structural topology optimization was by Jensen and associated researchers (Sandgren *et al.*, 1990 ; Sandgren and Jensen, 1992; Jensen, 1992). These studies involve the optimization of structures so as to minimize weight subject to stress and/or displacement constraints. This section will provide an overview of these studies, as well as some more recent extensions to them by Chapman (Chapman and Jakiela, 1994; Chapman *et al.*, 1993a, 1993b; Chapman, 1994) and by Kane (Kane *et al.*, 1995).

Jensen - Design Problems and Representations

The work of Jensen and associated researchers concentrated on three particular design problems: automotive decklid, bumper beam, and plane stress structural design. In the automotive decklid design problem, a three-dimensional sheet metal decklid is optimized to minimize weight subject to a maximum nodal displacement constraint. A rectangular representation of the shape of a decklid was used in order to simplify computations. This rectangular model was discretized into a 15x20 array of square elements. The design variables were the thicknesses of these elements, for which there were three possible values. Due to symmetry, only half of the elements were actually treated as variables.

In the bumper beam design problem, the goal was to optimize the cross-section of a beam, similar to that found in an automobile bumper. Again, the optimality criteria was minimum weight under a specified load with displacement always less than a given constraint value. The simplest problem considered a beam of uniform cross-section. A more complex case represented the beam as being made up of six elements, each with a potentially different cross-section. In both cases, the cross-section of a beam section was represented by a rectangular two-dimensional array of elements, as shown in figures 5.2 and 5.3. The design variables were the absence or presence of material in each element.

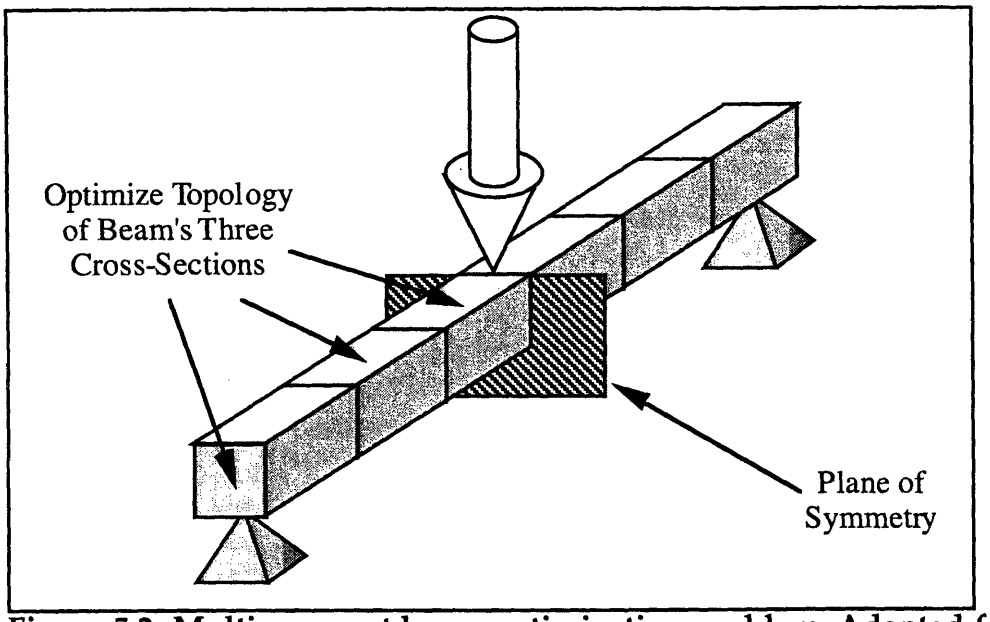


Figure 5.2 Multi-segment beam optimization problem. Adapted from Sandgren and Jensen (1992).

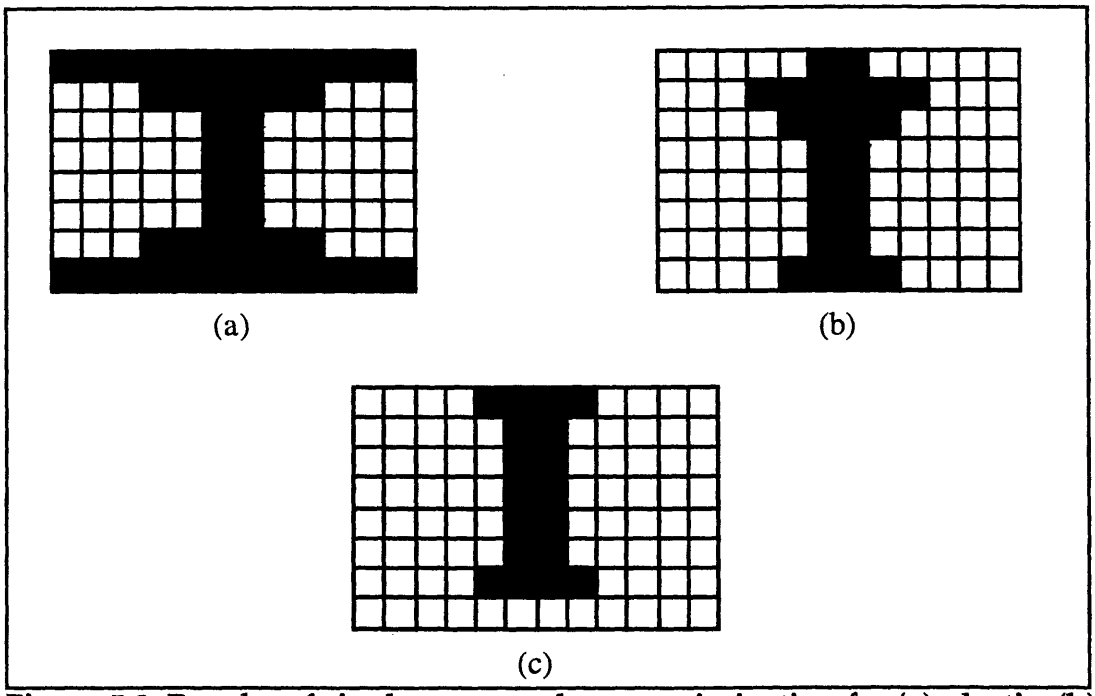


Figure 5.3 Results of single-segment beam optimization for (a) plastic, (b) aluminum, and (c) steel. Adapted from Jensen (1992).

The final problem involved finding an optimal topology for a plane stress structure, as shown in figure 5.4. The structure is subject to a known load, and the goal is to minimize the structure's weight while keeping the displacement at the point of load application less

than a fixed constraint value. Again, the representation used is that of an array of binary material/void elements.

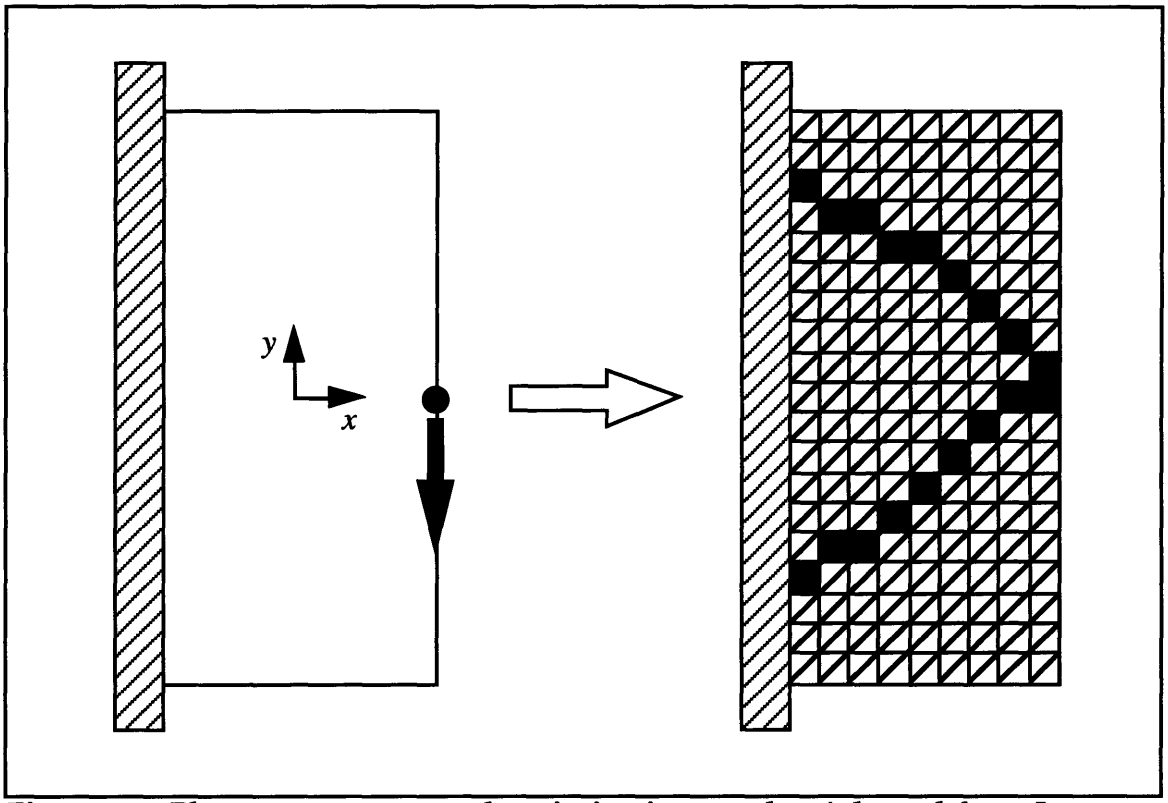


Figure 5.4 Plane stress structural optimization results. Adapted from Jensen (1992).

In all three cases, the design variables were represented using a 2-D array chromosome. In the decklid problem, each gene had three possible allele values (one for each possible thickness). In the other two problems, all of the genes were binary, indicating either the absence or presence of material in a given element.

Jensen - Optimization procedures

Similar optimization techniques were implemented in each case. In nearly every case, two-point crossover was used to "mate" two parent chromosomes. The only exception was the bumper beam made of multiple segments, each with an independent cross-section. In this case, a special version of one-point crossover was used. A "simple" GA with linear fitness scaling, roulette wheel selection and a randomly generated initial population was used in all cases.

Of course, different evaluation techniques were used in each case. In the decklid design problem, a commercial finite element package was used to perform the evaluations. Although the process was straightforward, the problem is difficult computationally, and approximately 30 seconds of computation were required to analyze one design. The plane stress problem also used finite element analysis to evaluate each structure, although this analysis was significantly quicker than in the decklid problem. Here, the difficulty came in devising a way to determine the stability of a structure. If the point of load application was not connected to a fixed point by solid material, the structure would be unstable and the finite element analysis would fail. To avoid this difficulty, the representation was changed slightly. Instead of an allele value of 0 indicating that no material was present, it was changed to mean that a material of relatively low stiffness was present. In this case, the "void" material was chosen to have a stiffness two orders of magnitude less than the "solid" material. This way, the finite element analysis would be capable of analyzing all structures, but unstable structures would receive very low fitness scores. The bumper beam case was the easiest case in terms of computation. The moment of inertia of each beam segment and the resulting beam bending characteristics could be computed analytically. The primary difficulty here was in determining what sorts of cross-sections were valid. For instance, one chromosomes could represent two separate topologies (see figure 5.5). This was allowed, and the beam segment was treated as one segment with a moment of inertia equal to the sum of the moments of inertia of the two topologies.

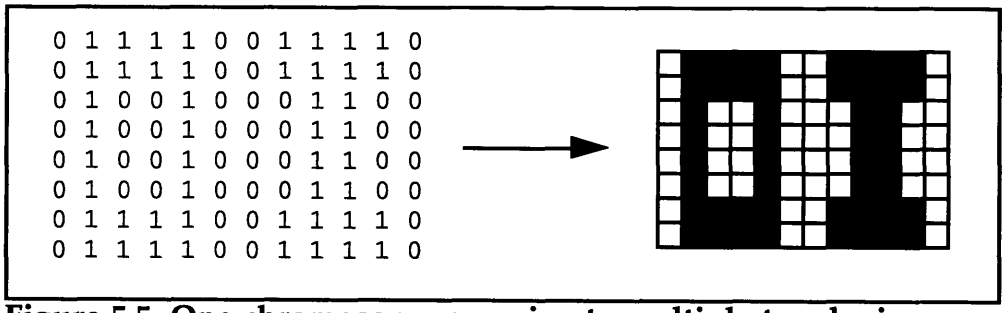


Figure 5.5 One chromosome mapping to multiple topologies.

Some parameters of the GA did vary from case to case. Most notable were the population size and number of generations. A parameter optimization technique was used to determine the "ideal" set of GA parameters. This optimization method tended to suggest very large population sizes (>150) and a relatively low number of generations (100-150). However, using such a large population for the actual problem would result in extremely long run-times. In general, the tendency of the researchers was to use as large a population as was feasible, in terms of computation time. In the computationally simplest problem, the

bumper beam optimization, a population of 800 individuals was run for 300 generations. In the plane structural optimization problem, a population of 140 was run for 150 generations. The difficulty of evaluating the fitness of a chromosome in the decklid problem resulted in a population limited to only 36 individuals being run for 150 generations. Despite this limitation, one run still took 48 hours to complete.

The work of Jensen and associated researchers demonstrated that genetic algorithms could be used to perform structural topology optimization with satisfactory results, except for a few drawbacks. In all cases, only one, "best," solution was obtained in each run. No attempts were made to determine whether multiple solutions existed with equally good fitness scores. In the decklid and plane structure problems, computation time was a significant issue, resulting in some GA parameters being set to non-optimal values.

Chapman - Design Problems and Representations

Similar structural topology optimization problems were investigated by Chapman and associated researchers. In many ways, this work is a direct extension of Jensen's work. The problems considered are the optimization of a beam's cross-section and the optimization of a cantilevered plate topology, similar to the plane stress structure investigated by Jensen. In each case, the design representation was basically the same as that used by Jensen, *i.e.*, a two-dimensional array of binary material/void elements. The only differences were that Chapman investigated the use of finer discretizations and used a lower value for the stiffness of a "void" element. In the case of the beam optimization problem, a 15x15 discretization was used, as opposed to the 6x8 discretization used by Jensen. Void elements were given a stiffness equal to 10^{-5} times the stiffness of material elements. Another difference from Jensen's work was that one-dimensional chromosomes were used by Chapman to represent potential designs (see figure 5.6). In this case, the values of the chromosome are "wrapped" around to form a 2-D array of values (the first five digits map to the first row, the next five map to the 2nd row, *etc.*). Finally, different fitness functions were used. While Jensen's work focused on minimizing weight and penalizing a structure for violating certain stress or displacement constraints, Chapman's work focused on maximizing the strength-to-weight ratio of a design, sometimes adding additional constraints as well.

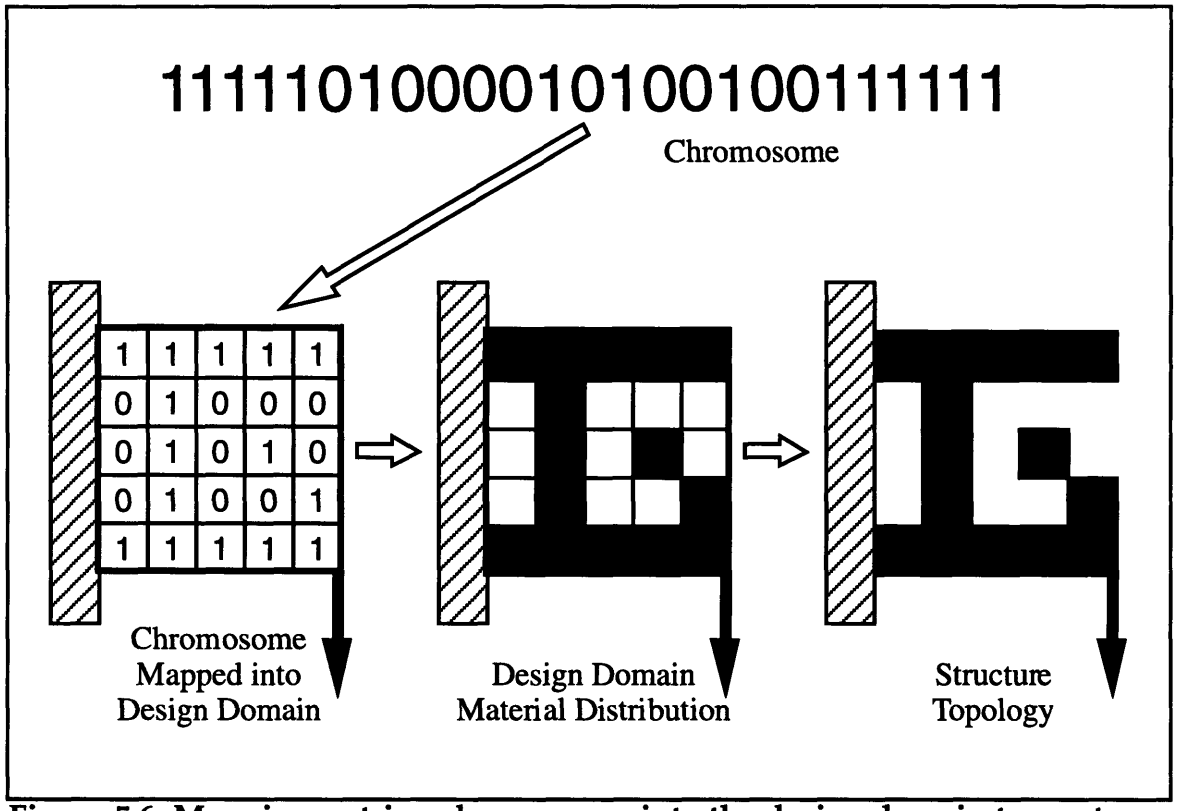


Figure 5.6 Mapping a string chromosome into the design domain to create a structure topology. Adapted from Chapman (1994).

Chapman - Optimization Technique

Greater differences exist in the techniques used to perform these optimizations, including Chapman's investigation of connectivity analysis, families of highly-fit designs, and manufacturing considerations.

Perhaps the most significant of these changes was the use of connectivity analysis, done to ensure that all structures generated were feasible. This connectivity analysis removed all material elements not connected to at least one "seed" element. In the case of the beam optimization, the seed element was chosen as the top center element. In the cantilevered plate examples, two seed elements were chosen along the fixed boundary, another was placed at the point of load application. Figure 5.7 shows an example of such a topology. Note that in order for an element to be "connected," it must share an edge boundary with another connected element. Material elements sharing only a corner connection are considered disconnected. This removal of disconnected material can be thought of as a specialized mapping of the design's genotype (the original, binary chromosome) to its phenotype (the information analyzed by the objective function). With connectivity analysis, disconnected material elements are given zero weight and are considered to be void when

performing finite element analyses. Without connectivity analysis, disconnected material elements are counted in weight calculations and are considered to be material when performing finite element analyses. It was empirically verified that this connectivity analysis improved GA performance (Chapman *et al.*, 1994).

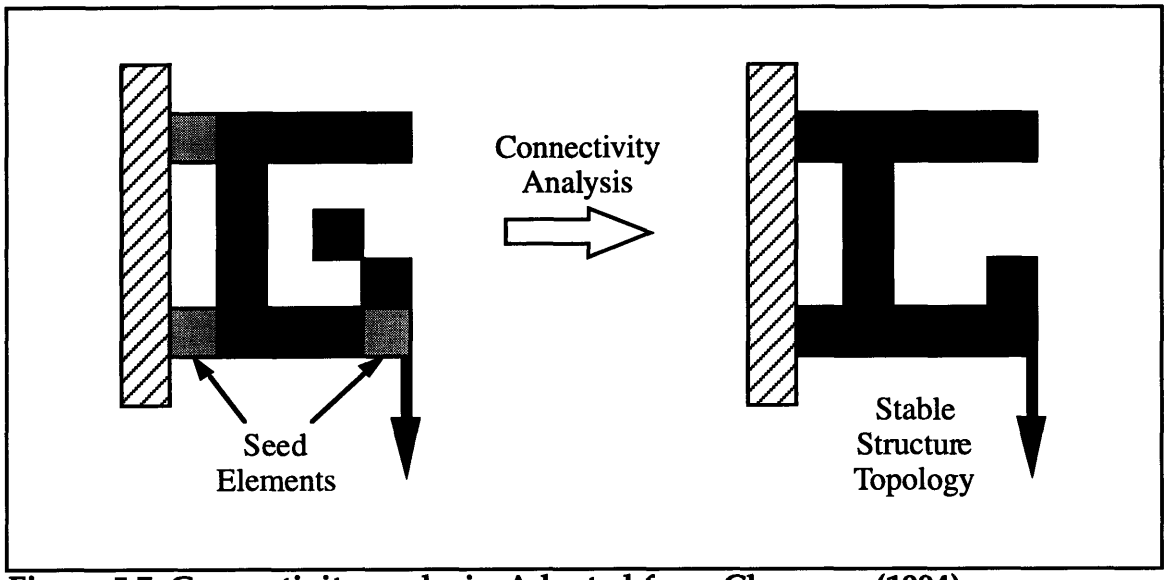


Figure 5.7 Connectivity analysis. Adapted from Chapman (1994).

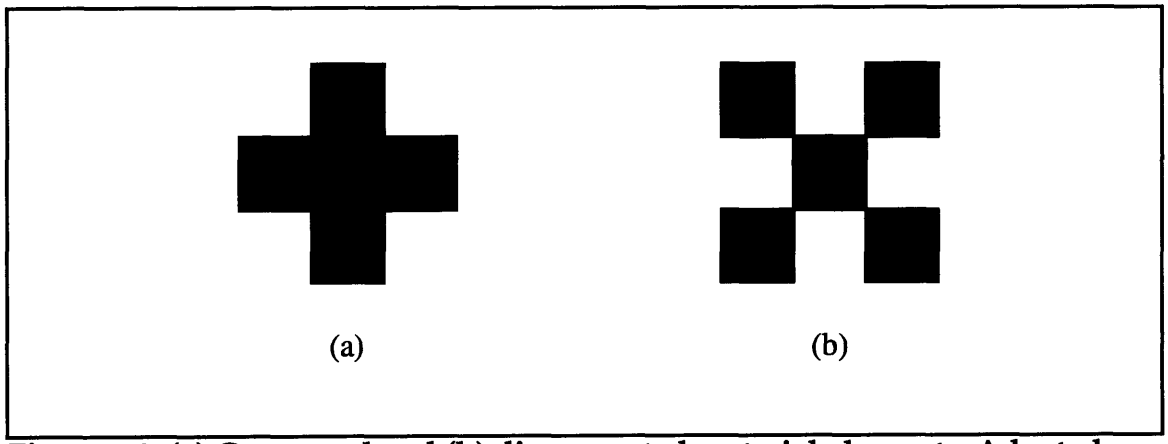


Figure 5.8 (a) Connected and (b) disconnected material elements. Adapted from Chapman (1994).

In the cantilevered plate example, the entire final population of the GA was treated as the "result." This is in contrast with most GA implementations which treat only the best individual as the result. By looking at the entire final population, some designs which are similar to the best overall design but slightly different are revealed. This allows the designer to use some secondary criteria, which might be difficult to model mathematically, when selecting a design. For example, Chapman investigated this idea using different fitness functions such as the one shown below.

$$Fitness = \frac{\left(\frac{stiffness}{weight}\right)}{perimeter} \qquad (5.2)$$

In this equation, a design's perimeter is equal to the sum of its outer perimeter and the perimeter of the internal holes. This fitness function was designed to encourage the generation of designs with high stiffness-to-weight ratios and little porosity. Some sample results are shown in figure 5.9. These are all individuals that were present in the final population of a single GA run. Note that although the designs are similar, each one has a different number of internal cavities. Although design d is the best in terms of stiffness-to-weight ratio, design a might be chosen since it would be easier to actually manufacture.

Topology	Number of Internal Holes	Stiffness-to-Weight Ratio	Displacement	Weight
(a)	0	1.19	0.0092	91
(b)	1	1.15	0.0102	85
(c)	2	1.11	0.0106	85
(d)	3	1.20	0.0094	89
(e)	4	1.14	0.0102	86
(f)	5	1.14	0.0101	87

Table 5.1 Plate topology performance data. Adapted from Chapman (1994).

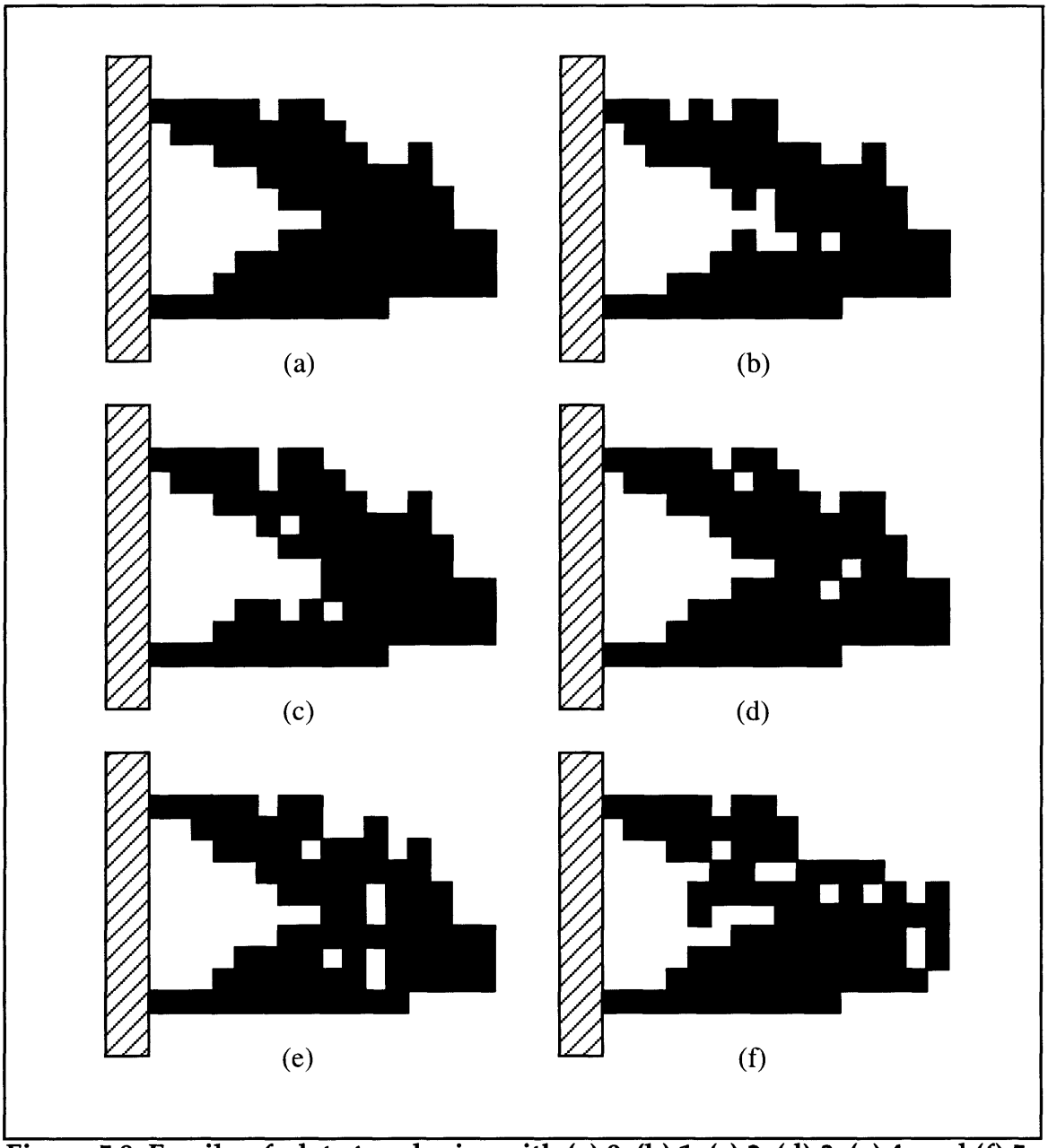


Figure 5.9 Family of plate topologies with (a) 0, (b) 1, (c) 2, (d) 3, (e) 4, and (f) 5 internal holes. Adapted from Chapman (1994).

Kane - Design Problems and Representations

Further work in structural topology optimization using genetic algorithms has been done by Kane *et al.* (1995). Their work examines the cantilevered plate optimization problem common to the work of Jensen and Chapman, with the goal of improving GA performance in terms of efficiency and diversity of solutions. The same 2-D array of material/void elements is used, although void elements are treated as being strictly void; no low-density

material is used. This results in slightly more accurate numerical results from the finite element analysis and the ability to consider loads applied to a structure's boundary (distributed loads caused by applied pressure, for example). However, this requires a new finite element mesh to be generated for each individual, increasing computation time. Two-dimensional array chromosomes are used, identical to those used by Jensen. A new type of mutation operator, epistatic mutation was also created in order to promote diversity in the GA population. Typically, mutation is performed by randomly selecting a bit and then flipping its value. Epistatic mutation selects bits whose value is nearly constant throughout the population in an effort to reintroduce the disappearing diversity. This new mutation operator was experimentally verified to outperform classical uniform mutation for this design problem.

Kane - Optimization Technique

The method of evaluation draws from the work of Jensen and from that of Chapman. Similar to Jensen's study, Kane uses a fitness function set up to minimize the weight of the structure subject to a constraint on the structure's displacement. Different results were obtained by varying the relative magnitude of the penalty term in this fitness function. Connectivity analysis is also implemented, as defined in Chapman's work, with some modifications. Kane's study uses only one "seed" element, placed at the point of load application. Structures not connected to the fixed boundary are given a fitness value of zero. This resulted in a greater diversity of solutions, as structures were not limited in where they could connect to the fixed boundary. Further examples by Kane use different fitness functions to compare the GA approach to homogenization-based techniques and to explore the effects of nonlinear geometrical effects caused by large deformations.

5.4.4 Summary

Overall, these investigations show that genetic algorithms can be an effective tool for performing structural topology optimization. Advantages of a GA-based approach include a great deal of flexibility in the types of design domains and fitness functions which may be used. The primary drawbacks are that such an approach requires a relatively large amount of computational power and a lack of diversity in the results. This investigation examines how to improve genetic algorithm performance in these regards, using design problems similar to those used by Jensen, Chapman, and Kane.

Chapter 6

This Investigation

6.1 Overview

This chapter first introduces the basic optimization techniques used in this investigation, describing how these techniques extend upon and differ from previous studies. Next, the design of planar stress structures by a speciating genetic algorithm using cluster analysis is detailed. Finally, a procedure for the design of adaptive structures using a hybrid genetic algorithm / simulated annealing approach is explained.

6.2 Optimization Technique

6.2.1 Introduction

Genetic algorithms are used to search for optimal structural topologies. An allowable design domain is discretized into a large number of binary material/void elements, yielding a combinatorial search space. Extending previous efforts with similar representations and search techniques, this investigation describes methods for improving genetic algorithm efficiency and for using a speciating genetic algorithm to distribute subsets of the evolving population of solutions over many local optima. This distribution of solutions is analogous to different species exploiting different niches in an ecosystem. Additionally, statistical cluster analysis techniques are used to quantify the extent to which a population is speciated, and this measure is used to probabilistically encourage mating of reasonably similar designs (*i.e.*, intraspecies mating).

6.2.2 Extensions of Related Studies

This research extends the work of Jensen, Chapman, and Kane (section 5.4.3) in genetic algorithm-based structural topology optimization, focusing on the optimization of two-dimensional plane stress structures. In contrast to the previous work in this field, this investigation includes the following:

Use of a genetic algorithm with overlapping populations

Definition of a new population initialization algorithm to start the GA with a population of "connected" structures.

Fitness sharing and speciation

Statistical cluster analysis used to limit the mating of dissimilar individuals

Introduction of a hybrid genetic algorithm / simulated annealing method for the design of adaptive structures

6.2.3 Similarities with Prior Investigations

The representation used in this investigation as well as some GA parameters and evaluation techniques are the same as some of those used in prior investigations. As in all previous studies of genetic algorithm-based structural topology optimization, the design domain is discretized into an array of binary, material/void elements. As in Chapman's investigations, void elements are given a weight and stiffness five orders of magnitude smaller than that of the material elements. Two dimensional array chromosomes were used in this study to represent potential designs, as was the case in the work of Jensen and of Kane. One-point crossover and simple mutation were used to "mate" two chromosomes. Connectivity analysis, the process defined by Chapman and also implemented by Kane to remove material elements that do not have a path of edge connections to a "seed" element, was also implemented in this investigation. The specific fitness functions used in this study were different from those in previous studies, although the goal of the optimization process was similar: to minimize weight subject to a displacement constraint.

6.3 Optimal Design of Planar Stress Structures

6.3.1 Introduction

In this study, genetic algorithms are used to perform structural topology optimization. The process of implementing a GA-based optimization technique can be broken into two main parts: choosing a representation of the design problem, and setting up the optimization process. Choosing a design representation involves some type of mapping of a particular design to a unique set of design variables. These design variables must then be encoded into some type of chromosome that can be operated on by the genetic algorithm. Setting up the optimization process involves defining a method for evaluating designs (a fitness function) and setting the GA parameters (population size, percent crossover, *etc.*) and

methods (parent selection method, fitness scaling, *etc.*). The following sections detail the representation and optimization process implemented in this investigation.

6.3.2 Design Representation

As in previous studies using genetic algorithms for structural topology optimization, a design is represented by an array of material/void elements. The design domain defines the maximum allowable extents of a structure. This domain is divided into an array of square elements, each of which can either contain solid material or be void. No intermediate densities are possible. Assigning values of "material" or "void" to each element defines one potential design. Figure 6.1 shows the process of going from a design domain to a structure's topology. Changing one or more of these design variables results in changing the structure's topology, and defines a new design.

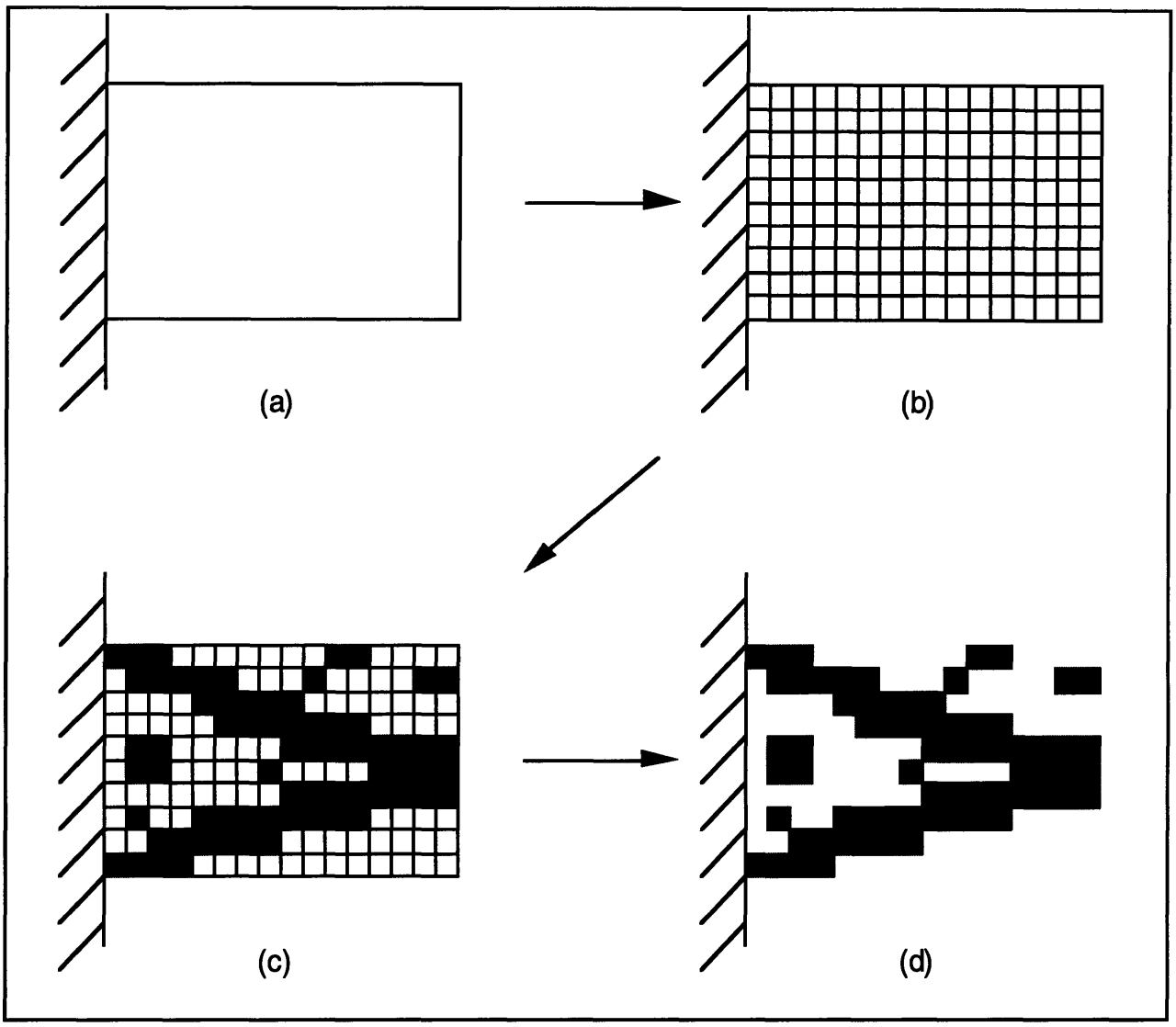


Figure 6.1 Design representation: (a) design domain, (b) discretized design domain, (c) design representation, and (d) structural topology (before connectivity analysis)

Design to genotype: Defining a chromosome

Such a topology must be represented as a chromosome, a format that the genetic algorithm can operate on. In this investigation, two-dimensional binary arrays are used as chromosomes. Each element of the array (each gene) represents the corresponding element in the design domain. A gene value of 1 means that material is present in the corresponding element of the design domain, a value of 0 means that the element is void, as shown in figure 6.2. This binary array is referred to as the genotype of the design. The result is a combinatorial search space of size 2^n, where n is equal to the total number of elements in the design domain. As a result, the size of the search space increases rapidly as finer discretizations are used.

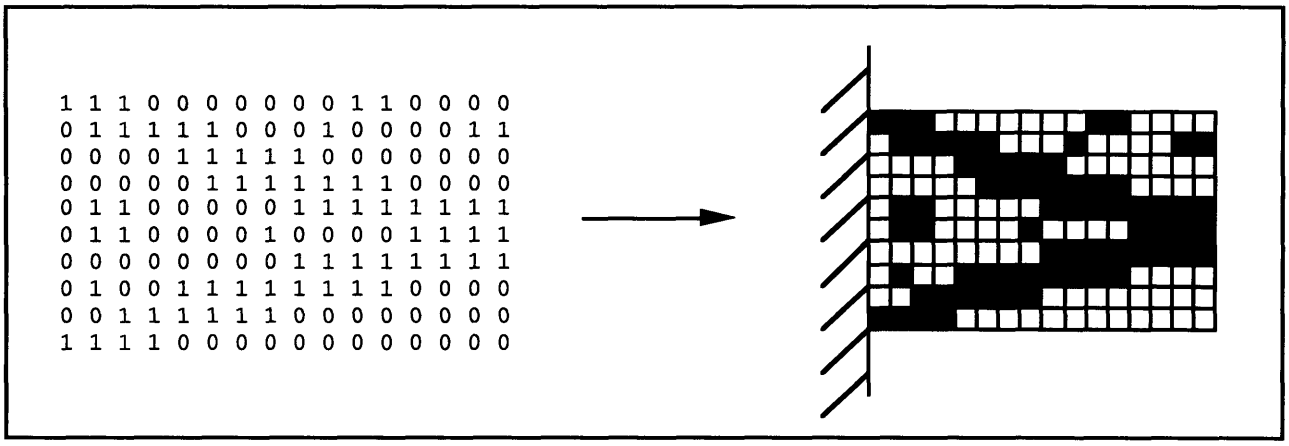


Figure 6.2 Mapping of a chromosome to the design domain

Genotype to Phenotype: Connectivity analysis

Connectivity analysis is used to map the genotype of a design into its phenotype, the representation analyzed by the fitness function. This mapping requires that all elements to be included in the structural analysis must be connected to other elements by at least one edge-edge connection, as opposed to only a corner connection (see figure 5.8). Also, constraints are made that material must be present in certain "seed" elements. All elements considered connected must be linked to one of these seed elements by a path of edge connections. Any elements not so connected are removed from the mesh for the purposes of structural analysis: they are switched to void elements. Since planar problems are being addressed, the rationale for connectivity analysis is that elements connected at corners cannot transfer moments about those corners (since the finite element model treats corners as kinematic pin joints). It is less likely, therefore, that "disconnected" elements will contribute to enhanced structural performance. This has been shown empirically (see specifically Chapman *et al.*, 1993, and Chapman's other preceding work for further discussion of this topic). Figure 6.3 shows the structure of Figure 6.2 after connectivity analysis. The topology that results from this connectivity analysis defines the design's phenotype, the chromosome itself (the genotype) does not change. This allows disconnected elements to in some sense be "recessive," in that crossover could cause them to combine with the (possibly disconnected) elements from another structure to yield a structure of connected elements of much improved performance.

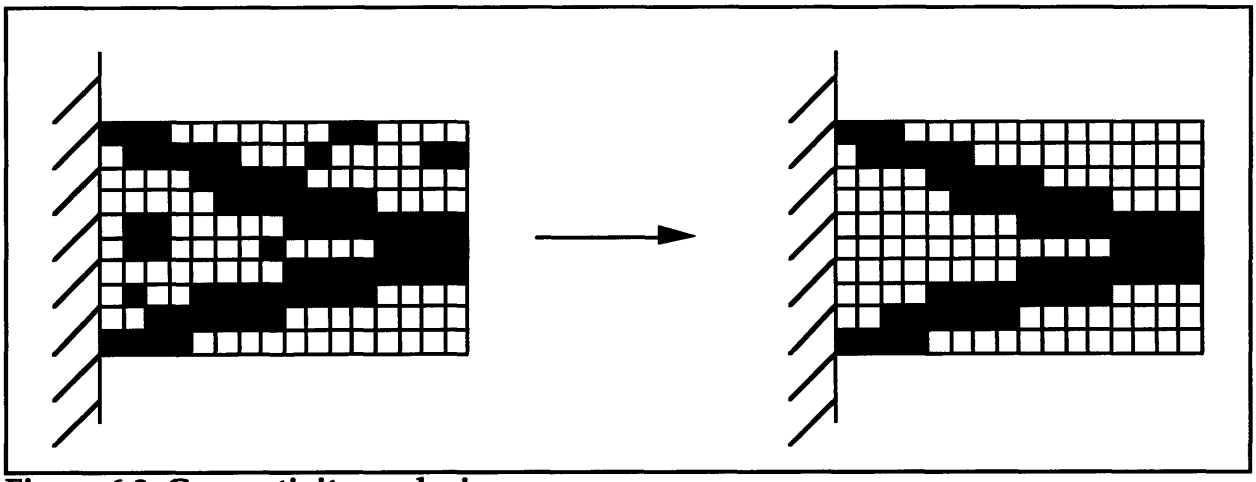


Figure 6.3 Connectivity analysis

6.3.3 Optimization Procedures

Once a design representation and chromosome mapping procedure have been decided upon, the details of the optimization procedure can be specified. Most importantly, a method for evaluating potential designs must be defined. This section explains the methods used to assign fitness scores to chromosomes and then to scale these scores so as to encourage diversity among the individual members of the GA's population. Also, the selection of various GA parameters and methods is discussed.

Structural Analysis: Phenotype evaluation

The primary aspect of fitness is structural performance. In this study, performance will be represented by two characteristics, mass and deflection. Generally, we wish the genetic algorithm to evolve structures that are both lightweight and deflect small amounts under a given load. The deflection will be measured with a finite element simulation, and the mass will be proportional to the number of connected material elements. These two characteristics can be used to create an unconstrained fitness, such as maximizing the following ratio:

$$fitness = \frac{1}{deflection * mass} \tag{6.1}$$

Alternatively, one of these characteristics can be optimized while treating the other as a constraint. For example, one might wish to minimize the mass of a structure subject to a

constraint on the maximum allowable stress in the structure. The specific fitness function used will be described with each example that follows.

As mentioned previously, each structure is evaluated using finite element techniques. After connectivity analysis has been performed, a finite element mesh representing the design domain is created. Each element of the design domain is represented by four triangular elements, as shown in figure 6.4a. This meshing technique is used for compatibility with existing finite element analysis code. Once this mesh has been defined, the nodes corresponding to support points (nodes on the seed elements along the fixed boundary) are constrained to have zero displacement, and concentrated loads are applied to the appropriate element nodes. Note that figure 6.4a shows that the entire design domain is meshed. As a result, the same mesh can be used for each individual design. Only the material properties of each element change from design to design. Creating a mesh for each individual, as was done by Kane and is shown in figure 6.4b, would yield slightly more accurate answers, but this re-meshing would require significantly more computation time per evaluation. A constant mesh technique was chosen for this study as it was believed that the small increase in numerical accuracy would not be worth the fairly large increase in computational expense. "Void" elements were given stiffness values 10^{-5} times that of solid material elements. This use of "soft" material is suggested by Bendsøe and Kikuchi (1988), who state that soft material can be considered a void or hole if its Young's modulus is 10^{-2} - 10^{-3} times that of a hard material. Jensen and Chapman both use similar techniques in their investigations to reduce computational expense. Once such a mesh has been defined and the constraints and loads have been applied, finite element analysis is performed. The results of this analysis are the resulting x and y displacements of each node in the mesh. Note that no stress calculations are performed. Analysis is performed using software written by Kumar (1993) for analysis of planar topologies.

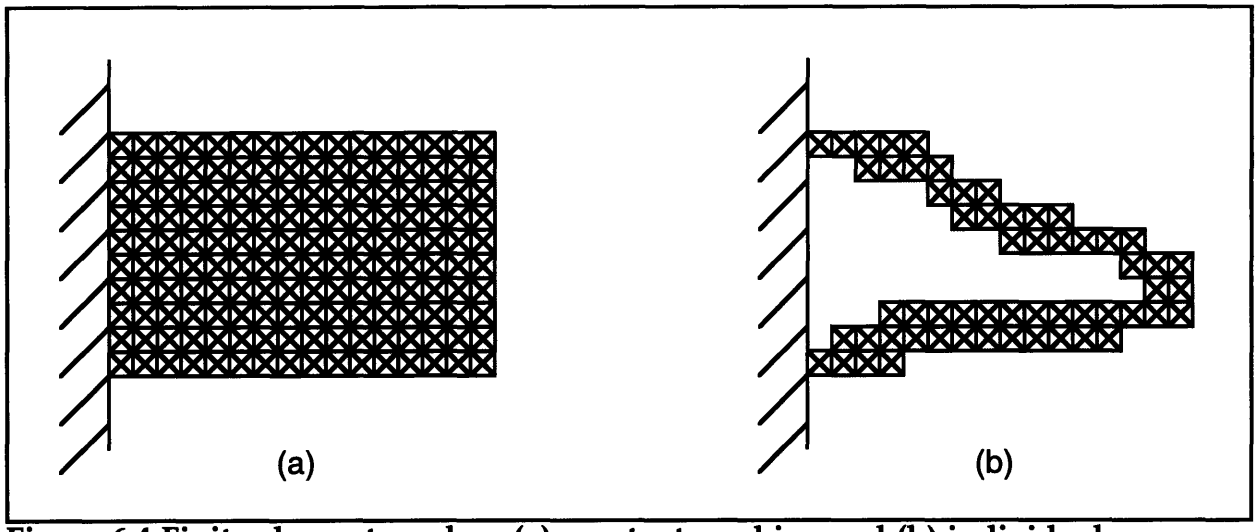


Figure 6.4 Finite element meshes: (a) constant meshing and (b) individual meshing

Fitness sharing and speciation

Fitness sharing is implemented to encourage diversity in the GA population. As described in chapter 2, this involves individuals sharing their fitness with other similar individuals. Implementing such a system requires that some measure of similarity between individuals be defined. The similarity function chosen for this investigation, called the Jaccard Coefficient (see Romesburg, 1984, page 143), is applied to the phenotype designs (the designs *after* they have undergone connectivity analysis). The "similarity" between two designs is defined as follows. Comparing two material/void design arrays on an element-by-element basis, first determine the following intermediate parameters:

a = number of corresponding elements that are both material
b = number of corresponding elements where exactly one is material

The Jaccard similarity coefficient is then computed as follows:

$$C_j = \frac{a}{a+b} \tag{6.2}$$

There are two things to note about this metric. First, it does not take into account void/void matches. As evolutionary optimization progresses, considerations of minimizing weight will lead to relatively sparse assignments of material to the allowable domain for most of the problems addressed here. Also, considering these matches in (6.2) would not provide additional discrimination. Second, the coefficient ranges from a value of zero for no material/material matches to one for an identical material/material match over the domain.

As sharing functions require a distance function, not a similarity function, the following conversion is used:

$$d = 1 - C_j \tag{6.3}$$

In this case, two identical structures have a distance of zero, and two structures with no material elements in common have a distance of 1. Thus, it agrees with the required behavior of a distance function as described earlier. Results obtained using this distance function and a variety of sharing functions are presented in the following chapter.

This distance function is also used to recognize the formation of species during the evolution. This information is then used to restrict mating between excessively dissimilar chromosomes. This is done by using cluster analysis techniques (see Romesburg, 1984) to partition the population into subsets of similar chromosomes at each generation. To understand how this is done in the context of (dis)similar structural topologies, consider figure 6.5, which shows an abstraction of a structural topology space. This is shown as a two-dimensional graph only for explanatory purposes, in reality the structural topology space is n-dimensional, where n is equal to the number of elements in the design domain discretization. Dots on the graph represent topologies, with the distances between the dots analogous to the amount of dissimilarity or distance between the topologies.

The first step in this clustering process is to compute the "resemblance," or similarity, between each pair of topologies. This is done by computing the Jaccard coefficient for each pair as described above. If there are n topologies, these values can be stored in an n x n lower triangular matrix called the resemblance matrix. The pair that is most similar (*i.e.*, highest coefficient value) is "clustered" and treated as a single entity as long as its similarity is above a chosen threshold. As shown in figure 6.5b, points 1 and 5 are the first pair to be clustered. A new (n-1) x (n-1) resemblance matrix can now be computed, with "1-5" considered a single point for subsequent clustering purposes. The distance between a new point, x, and 1-5 is defined as the average distance between the new point and 1 and the new point and 5. Continuing with a second iteration of the process, points 2 and 3 are now most similar with similarity above the threshold, so they are clustered as shown in figure 6.5c. The next iteration clusters 4 with 1-5, computing a clustering similarity as the average of the 4-1 and 4-5 similarities, as shown in figure 6.5d. The final iteration (figure 6.5e)

clusters 6 with (2-3) in a like manner. The process is now complete since 4-(1-5) and 6-(2-3) will not cluster because their similarity is below the threshold.

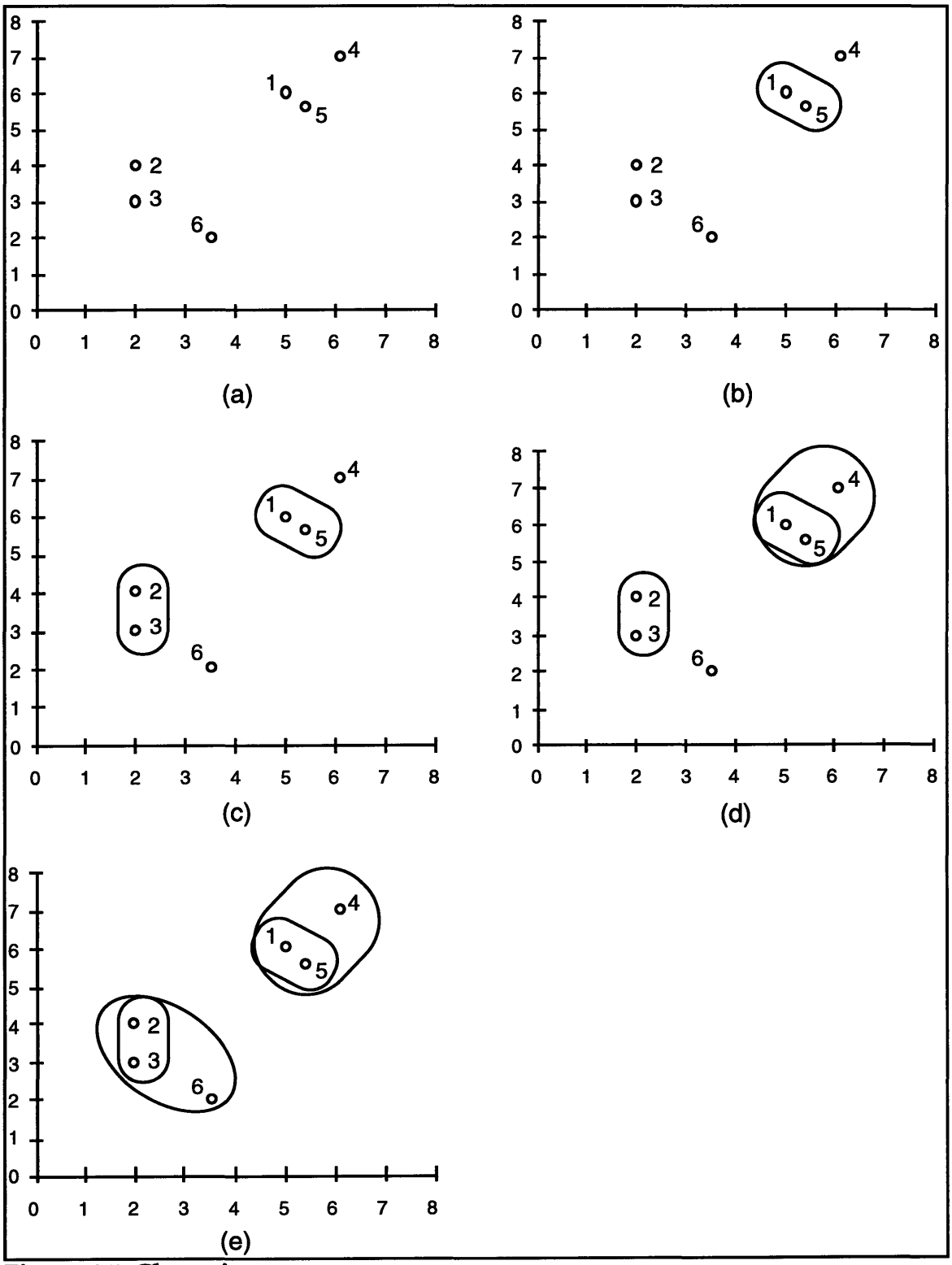


Figure 6.5 Clustering process

For a given set of data (in our case a given population of topologies), it is possible to measure the "tightness" of a clustering by comparing the actual similarities between pairs of points with the similarities that were used in the clustering process. These similarities differ, as shown above, when clusters are treated as a single point in the computation of the Jaccard coefficient. Our example of figure 6.5 has six points, yielding fifteen possible point pairs, as shown in table 6.2:

Point Pair	"X" Actual C_j	"Y" Clustering C_j
(1,2)	$C_{j(1,2)}$	*
(1,3)	$C_{j(1,3)}$	*
(1,4)	$C_{j(1,4)}$	$C_{j(4,(1-5))}$
(1,5)	$C_{j(1,5)}$	$C_{j(1,5)}$
(1,6)	$C_{j(1,6)}$	*
(2,3)	$C_{j(2,3)}$	$C_{j(2,3)}$
(2,4)	$C_{j(2,4)}$	*
(2,5)	$C_{j(2,5)}$	*
(2,6)	$C_{j(2,6)}$	$C_{j(6,(2-3))}$
(3,4)	$C_{j(3,4)}$	*
(3,5)	$C_{j(3,5)}$	*
(3,6)	$C_{j(3,6)}$	$C_{j(6,(2-3))}$
(4,5)	$C_{j(4,5)}$	$C_{j(4,(1-5))}$
(4,6)	$C_{j(4,6)}$	*
(5,6)	$C_{j(5,6)}$	*

Table 6.1 Point pair resemblances, actual and clustering

The "*" in the "Y" column indicates that the point pair was not clustered, so there is no clustering similarity. This allows us to abbreviate the table as follows:

Point Pair	"X" Actual C_j	"Y" Clustering C_j
(1,4)	$C_{j(1,4)}$	$C_{j(4,(1-5))}$
(1,5)	$C_{j(1,5)}$	$C_{j(1,5)}$
(2,3)	$C_{j(2,3)}$	$C_{j(2,3)}$
(2,6)	$C_{j(2,6)}$	$C_{j(6,(2-3))}$
(3,6)	$C_{j(3,6)}$	$C_{j(6,(2-3))}$
(4,5)	$C_{j(4,5)}$	$C_{j(4,(1-5))}$

Table 6.2 Clustered point pairs

Intuitively, the tightness of the clustering is related to the differences of corresponding values of the X and Y columns: how much was the data "distorted" in order to consider it clustered? An aggregate value for this notion is provided by the Pearson product-moment correlation coefficient,

$$r_{x,y} = \frac{\sum xy - \frac{1}{n}(\sum x)(\sum y)}{\{[\sum x^2 - \frac{1}{n}(\sum x)^2][\sum y^2 - \frac{1}{n}(\sum y)^2]\}^{\frac{1}{2}}} \qquad (6.4)$$

During an evolutionary optimization, the value of $r_{x,y}$ is monitored to determine the extent to which sharing-based speciation has partitioned the population. Note that initial randomly generated populations can be very well speciated (in a non useful manner) if the chromosomes are mutually dissimilar. $r_{x,y}$, therefore, typically drops in value in the early generations and then grows to a higher value. When niche formation has sufficiently progressed, interspecies mating is increasingly unlikely to produce improved designs. This fact is taken into account by using $r_{x,y}$ as another probabilistic factor on the occurrence of mating. After mating restriction is started, $r_{x,y}$ is computed with a threshold value of C_j. Each time a pair of parents is selected, there is an $r_{x,y}$ % chance that this pair must be in the same species (defined by C_j). If this probabilistic test fails, the pair is discarded and a new pair is chosen for testing. In this way, the likelihood of interspecies mating decreases as $r_{x,y}$ increases. A typical threshold value for C_j is 0.85. In our tests, we have initiated mating restriction after a preset number of generations (halfway through a run). Over the second half of a run, $r_{x,y}$ typically ranges in value from 0.70 to 0.90.

6.4 Design of Adaptive Structures

6.4.1 Introduction

This section introduces a hybrid genetic algorithm / simulated annealing method and suggests a methodology for using it to perform optimal design of adaptive structures. The goal is to optimize the design of a structure whose geometry is capable of changing between two different states. Specifically, the cross-section of a beam is optimized for performance under two sets of loading conditions, pure bending and pure torsion. In this study, a genetic algorithm is used to generate potential structures. The fitness function first evaluates each structure's performance under an applied load causing only a bending moment, then uses simulated annealing to see how well the structure can "adapt" (change its topology) in response to a change in operating conditions (a torsional load replaces the bending load). The total fitness of a structure is a combination of the fitness of the original and the "adapted" topologies. As in the previous case of designing planar stress structures, a discretized design domain is used, and only two-dimensional optimization is performed.

6.4.2 Design Representation

Defining a chromosome

The goal of this optimization process is to optimize the topology of a structure with a uniform cross-section. As in the problems described above, a design is represented by an array of binary, material/void elements, as this representation is appropriate for use with both genetic algorithms and simulated annealing. In this case, unlike previous examples, the design domain represents the cross-section of a structure (see figure 6.6). This representation is similar to that used by Jensen for the bumper beam design optimization problem (refer back to figures 5.3 and 5.5), and that used by Chapman for a similar problem (Chapman, 1994). In this investigation, a two-dimensional binary array is again used as a chromosome for the genetic algorithm. Similar to Chapman's work, one seed element (an element in which material *must* be present, shown in black in figure 6.6) is chosen and connectivity analysis is performed to eliminate disconnected material. This differs from Jensen's work, in which multiple cross-sections could be present in a single design domain (see figure 5.5).

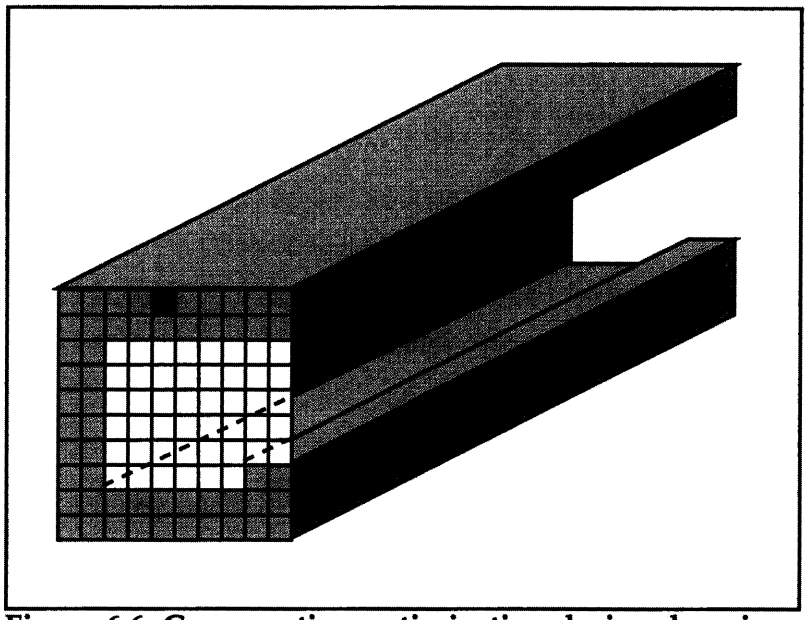


Figure 6.6 Cross-section optimization design domain

Modeling an adaptive structure: neighborhood operators

The neighborhood operators used by the simulated annealing process are used to represent the way in which a structure adapts to change in conditions. As described in section 5.3, simulated annealing uses a neighborhood operator to transform an existing design into a similar but slightly different structure. In this study, this operation is meant to simulate the effect of an actuator embedded into the structure, capable of causing a change in the structure's topology. Two sample neighborhood operators are illustrated in figure 6.7 below.

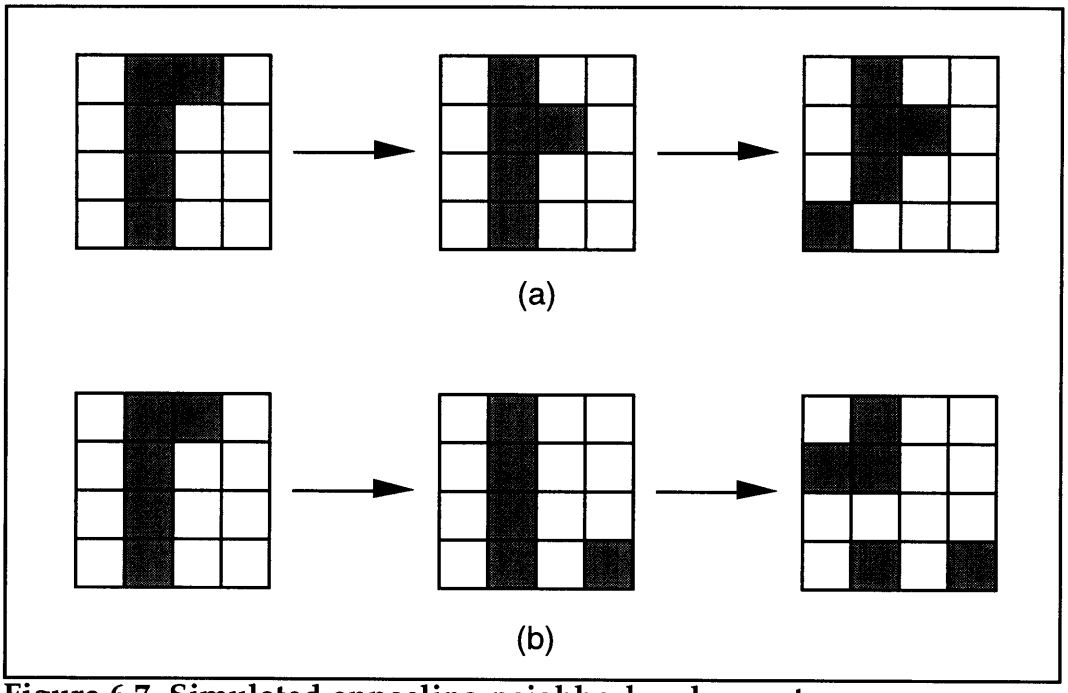


Figure 6.7 Simulated annealing neighborhood operators

In the neighborhood operator shown in figure 6.7a, one material element is chosen at random, then moved to one of its neighboring void elements. If the selected element is surrounded by material, another element can be selected. The move may be restricted to elements with edge connections (up, down, left, or right), or diagonal moves may be allowed. A restriction may also be made that the moving element must still be connected to another material element after the move. In this case, the first move shown in figure 6.7a would be allowed but the second one would not.

Figure 6.7b shows another possible neighborhood operator, based on genetic algorithm mutation. In this case, two elements are randomly selected, one material and one void, and reversed (the material element becomes void and vice-versa).

In practice, selection of a neighborhood operator would be dependent on the type of actuators being used. Ideally, a good neighborhood operator would produce structural changes similar to those caused by the actuators in the actual structure.

6.4.3 Optimization Procedures

In order to optimize structures capable of changing topology, this method combines the use of genetic algorithms and simulated annealing. A GA is used to generate potential designs,

and then a simulated annealing process is used to measure how well each design can "adapt" to a change in loading conditions. This section gives a detailed explanation of this process.

Genetic algorithm approach

A steady-state genetic algorithm is used to evolve a population of potential structures. A randomly generated initial population of structures (represented by binary array chromosomes as described in the previous section) is created, and then the process of evaluation, selection, and reproduction begins. Selection and reproduction of chromosomes are performed using the same GA techniques as in the previous example (roulette wheel selection, one-point crossover, *etc.*).

Evaluation of a topology's performance under one set of loading conditions is done analytically; no finite element analyses are performed. In this case, the only computations made are to find the mass properties of the cross-section (mass, location of the center of mass, and moments of inertia). First, connectivity analysis is done to remove all material not connected (again, only edge-edge connections are considered) to the "seed" element. The mass of the cross-section (number of material elements) is then computed. Next, the location of the center of mass of the cross section is calculated. Finally, the rectangular and polar moments of inertia of the cross-section about its center of mass are computed (I_x, I_y, and J_O, as shown in figure 6.8). Because no finite element analyses are required for these calculations, they can be performed very rapidly and require little computation time. Also, since no meshing needs to be done, void elements are assigned a density and stiffness of zero. As in the previous case, the fitness function may combine the mass properties to perform an unconstrained optimization (maximize J_O/mass, for example). Alternatively, one or more properties may be treated as constraints (minimize mass while keeping $I_x > I_{min}$).

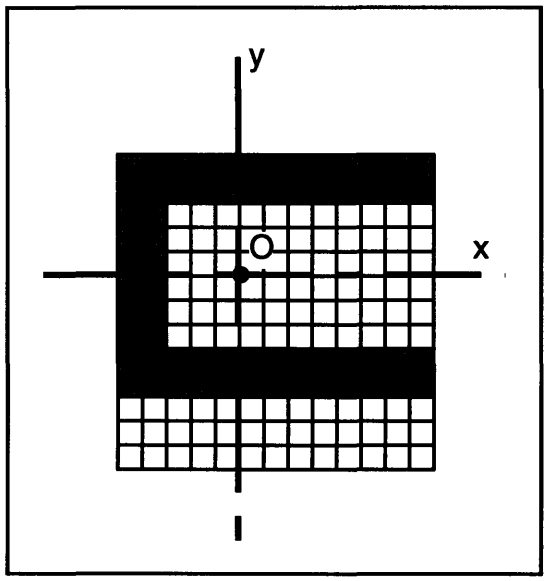


Figure 6.8 Sample design domain

Simulated Annealing: letting a structure adapt

A simulated annealing process is used to allow a structure to change its topology to achieve better performance under different loading conditions. As described above, a genetic algorithm is used to generate a population of structures. First, each of these structures is evaluated under a given load (a bending moment about x, for example). Then, this load is removed, a new one is applied (a torsional load about point 0, for example), and the performance is measured again. Next, a simulated annealing process is performed to optimize the topology for performance under this second loading condition, simulating the ability of the structure to adapt itself based on changes in its environment. Simulated annealing may be performed either for a fixed number of steps or until the design reaches some predefined level of performance. In the first case, the final structure's performance is recorded. In the second case, the number of steps required to reach the desired performance level is recorded. Because successive runs of this simulated annealing process from the same initial design may produce significantly different results, the process is repeated ten times for each topology and an average value of the results is computed.

The fitness function of the GA then combines the fitness of the original topology with the fitness of the "annealed" topology to determine the overall fitness of the design. In this way, the GA is capable of evolving structures which perform well given one set of loading conditions and can adapt so as to also perform well under a significantly different type of load. Specific examples of fitness functions are given in the following chapter.

Chapter 7

Results:

Design of Planar Stress Structures

7.1 Overview

This chapter presents examples of structural topology optimization performed using genetic algorithms as described in the previous chapter. Topology design of planar stress structures is examined. The effects of various parameters and methods are investigated, with a focus on GA efficiency (speed) and final population diversity.

7.2 Example 1: Cantilevered plate

This example considers the generation of topologies for a cantilevered plate subjected to a vertical load. The design domain for this example is the 10 x 16 discretization shown in figure 7.1, and is the same domain that has been used in some previous studies (Chapman *et al.*, 1994; Chapman and Jakiela, 1995). The two elements shown in black on the left of figure 7.1 represent the support points. The nodes along the left side of these elements are constrained to have zero displacement in the FEM analysis. A point load is applied to the element shown in black on the right, to the FEM node in its lower right-hand corner. The final output of the finite element analysis is the magnitude of this node's displacement.

Material properties for this example were those of 1 mm thick steel (E = 200 GPa, ν= 0.3). The size of each element in the design domain is 1 cm x 1 cm.

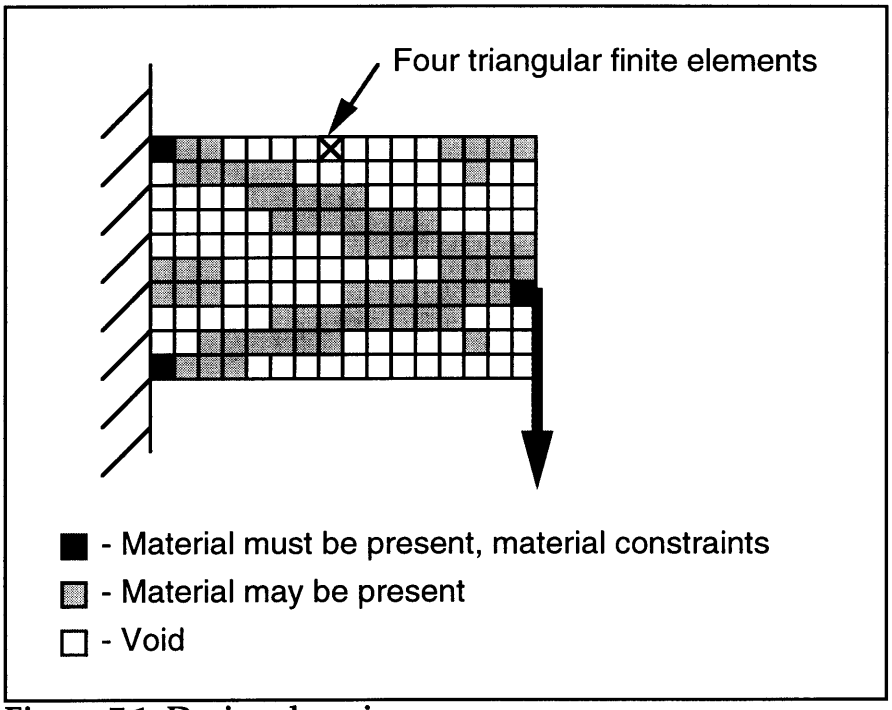


Figure 7.1 Design domain

In this example, the goal of the optimization is to produce topologies of minimal mass while keeping the displacement (δ) at the point of load application less than a specified amount (δ_{max}). The fitness of a chromosome is computed as follows. First, connectivity analysis is performed to remove disconnected material. Then, a normalized mass is computed as the percent of elements in which material is present. For instance, if a structure in this 10x16 discretization had 40 elements of material present (and, therefore, 120 elements of void), its normalized mass would be 0.25. The connectivity analysis also checks to see if the point of load application is connected to the fixed nodes by material. If the loaded element is not connected to either fixed node, the structure's fitness is given by (7a); if it is connected to only one fixed node, (7b) is used. In these cases, no structural analysis is performed: the finite element analysis is only done for structures in which the load point is connected to both fixed nodes. For such structures, the fitness score is calculated according to (7c) if the displacement constraint is violated and by (7d) if it is not. These equations provide the raw fitness score of a chromosome.

$$\text{fitness} = 5*\text{mass} \tag{7a}$$

$$\text{fitness} = 50*\text{mass} \tag{7b}$$

$$\text{fitness} = 1000 - 500*(\delta - \delta_{max}) \tag{7c}$$

$$\text{fitness} = 1100 - \text{mass}*100 \qquad (7d)$$

For this example, the magnitude of the applied load was set to 4250 N and the allowable displacement (δ_{max}) was 0.01 m.

7.2.1 Effects of a New Population Initializer

Like most genetic algorithms, the first runs all used an initial population which was generated randomly. As a result, several of the chromosomes in the initial populations did not represent connected structures. On average, less than 1% of all randomly generated chromosomes represent feasible (*i.e.*, connected) structures for this design problem. It is expected that as the design domain becomes more finely discretized, this process of finding connected structures would become more and more difficult. Because the GA started out with such a poor set of initial chromosomes, the early generations of the GA run were spent finding solutions that were merely feasible, instead of searching for an optimum. In order to avoid this inefficiency, a preprocessor was used to generate an initial population consisting of connected structures only. This was achieved using a separate genetic algorithm. The pre-processor GA started with a random initial population and used the following parameters and fitness function:

GA Type:	Simple GA	
P_{cross}:	0.95	
P_{mut}:	0.01	
Population size:	30	
Fitness =	0 + 100*(connected mass)	(7e)
	500 + 100*(connected mass)	(7f)

If a structure is not connected to either fixed point, equation (7e) is used to compute its fitness. If the structure is connected to just one of the fixed points, equation (7f) is used. In each case, the mass of the structure is calculated *after* connectivity analysis has been performed and is expressed as a percentage of total possible mass. For example, a structure with 40 elements connected (out of 160 possible) would be assigned a mass of 0.25. These equations were designed to give higher fitness scores to structures that are "closer" to being connected to both fixed points. A structure with more mass is likely to require fewer changes to make it fully connected than a lighter structure would. If the structure is connected to both fixed points, the GA run is halted and that structure is recorded as one member of the initial population for the main GA run. In this case, an "optimal" structure is

simply one that meets the requirement that it connects the point of load application to both fixed points with solid material. Although measuring the mass of a structure may not be the best way to encourage connectivity, it was found to work reasonably well, and the GA was quick to find connected structures.

As described above, this preprocessor GA would run only until it found one connected structure, and this individual would be saved and used as one member for the initial population of the main GA. Thus, the preprocessing GA was run a number of times, n, equal to the population size of the main GA. A typical runtime for the preprocessing GA was approximately 1 sec. Thus, creating an initial population of sixty connected structures could be completed in about one minute, a time very small compared with the runtimes of the main topology optimization genetic algorithm. Figure 7.2 and table 7.1 show the improved results obtained using this preprocessor GA to generate an initial population of connected structures. In each case, the GA was able to find a structure meeting the displacement constraint. All results shown are averages taken from five GA runs. New populations were generated for each run, but all other parameters and methods were constant for each of the five.

It is important to note that the time required to run the GA preprocessor with a population size of 60 (~ 1 min.) is about the same magnitude as the time taken for 8 generations of the main GA (1 min. $\cong$ 8 * 30 minutes/run / 250 generations). From the results shown below, we can see that the time is better spent generating a connected initial population (*i.e.*, little improvement is made by running the GA for 8 generations). Also, it is evident that using a connected initial population does not have an adverse effect on population diversity. Population diversity is calculated by computing the distance (0 to 1) between each possible pair of chromosomes in the final population, then taking the average of all of these values.

It was found that using the preprocessor genetic algorithm to generate an initial population of connected structures resulted in better optimization performance while maintaining similar runtimes and population diversity. It is important to note that although the GA began with a population of connected structures, crossover and mutation could still result in child chromosomes representing disconnected individuals. As a result, this preprocessor GA was used for all examples that follow.

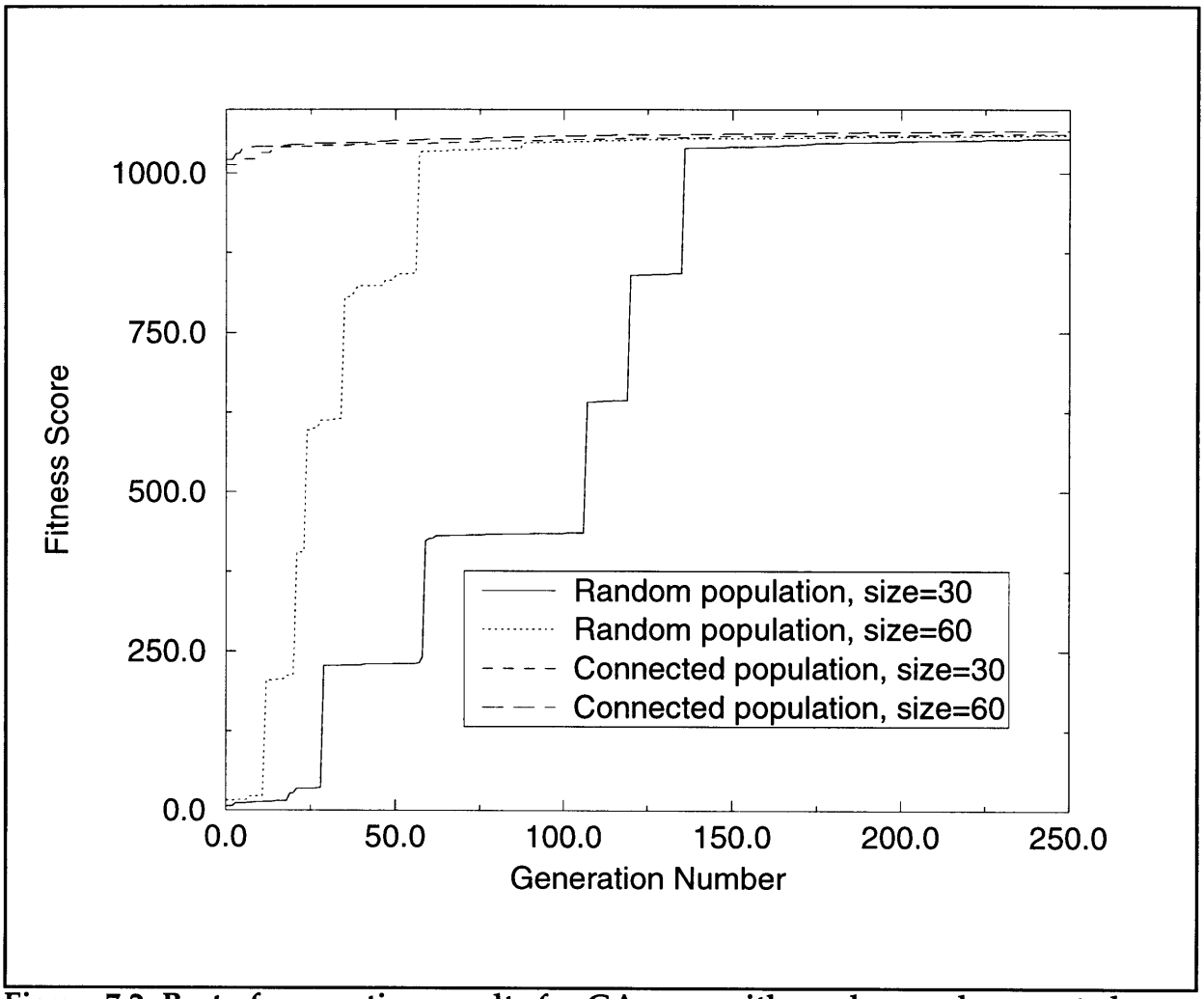


Figure 7.2 Best of generation results for GA runs with random and connected initial populations (of size 30 and 60)

	Best Fitness Score	Mass	Population Diversity	Runtime (minutes)
Random, popsize=30	1054	46%	11%	9±4
Random, popsize=60	1059	41%	15%	25±3
Connected, popsize=30	1061	39%	11%	15±2
Connected, popsize=60	1063	37%	15%	30±3

Table 7.1 Best of generation results for GA runs with random and connected initial populations (of size 30 and 60)

7.2.2 Effects of Connectivity Analysis

Next, runs were done to confirm the value of performing connectivity analysis *during* a run. First, runs were done using no connectivity analysis at all (*i.e.*, no material was removed from any of the chromosomes during the runs). Next, runs were done using a relaxed version of connectivity analysis, in which corner-to-corner connections were allowed (*i.e.*, two elements sharing one common node are considered connected). Finally, runs were done using full connectivity analysis (two elements must share two nodes to be considered connected). In each case, the initial population (of size 60) was generated according to the type of connectivity analysis being used. The fitness function used was that of equations (7a)-(7d). Of course, in the case where no connectivity analysis was performed, equations (7a) and (7b) were not used. The results of these runs are shown in figure 7.3 and are summarized in table 7.2 below. Again, all figures are averages taken from a set of 5 GA runs.

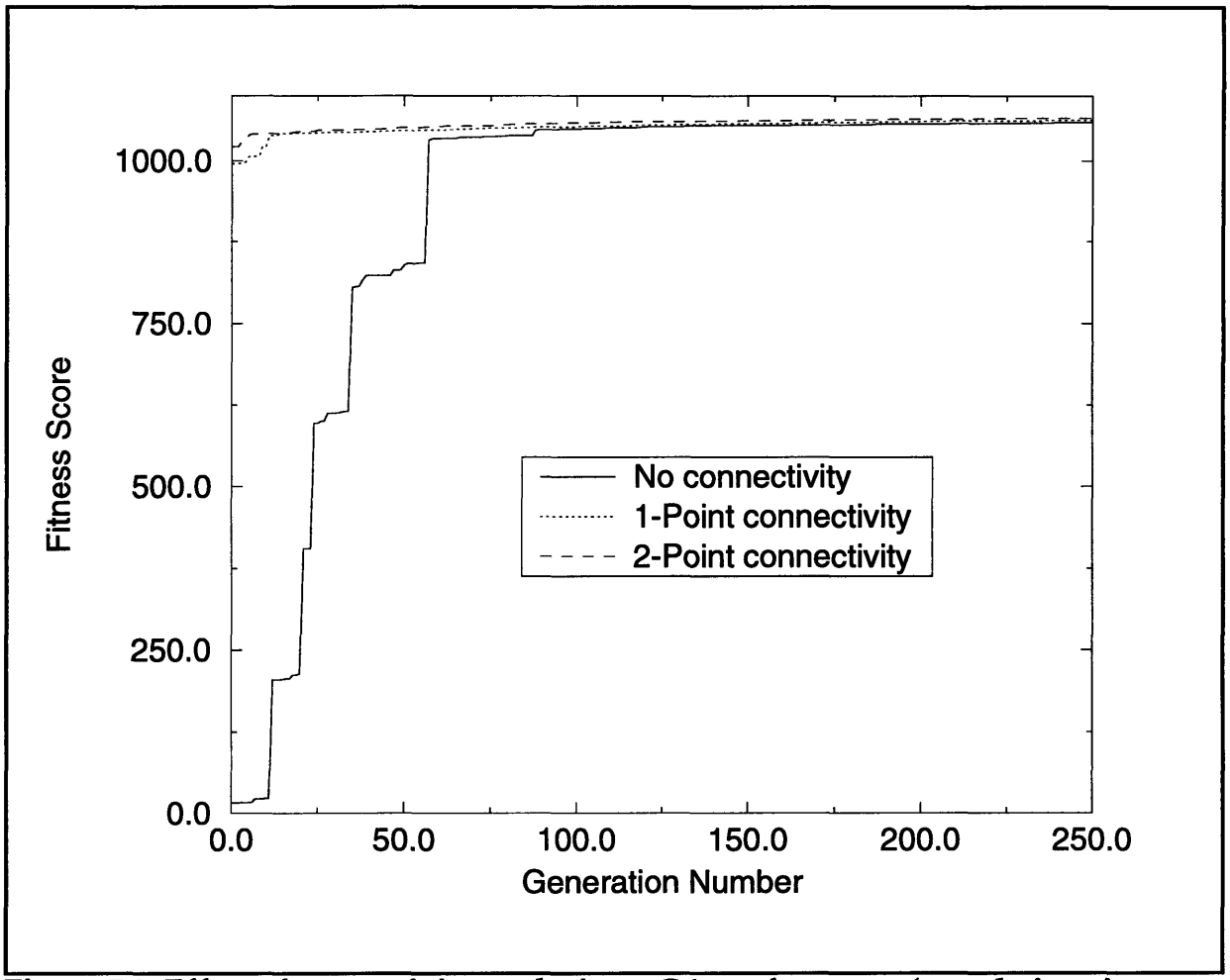


Figure 7.3 Effect of connectivity analysis on GA performance (population size = 60)

	Best Fitness Score	Mass	Population Diversity	Run Time
No Connectivity	1059	41%	15%	46±5
One-Point Connectivity	1060	40%	14%	45±5
Two-Point Connectivity	1063	37%	15%	31±5

Table 7.2 Effect of connectivity analysis on GA performance

From these results, it is clear that connectivity analysis is useful for improving results of a GA run and reducing the time taken to perform such a run. Eliminating connectivity analysis altogether results in much longer runtimes, as finite element analysis must be performed on every individual. Figure 7.4 shows some typical results obtained from runs using (a) no connectivity analysis, (b) one-point connectivity analysis, and (c) two-point connectivity analysis. Each structure pictured was the best individual found by a single GA run. All further examples use two-point connectivity analysis.

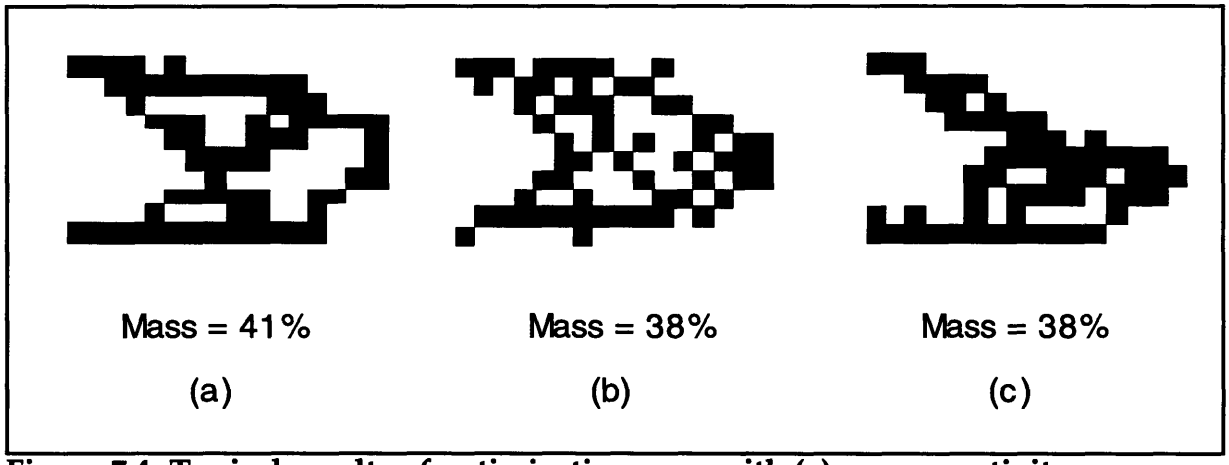


Figure 7.4 Typical results of optimization runs with (a) no connectivity analysis, (b) 1-point connectivity analysis, and (c) 2-point connectivity analysis

7.2.3 Fitness Sharing

Although the techniques implemented in the previous examples were successful at improving GA performance in terms of the best overall individual found, they did not result in any significant increases in population diversity. The following set of GA runs use fitness sharing (as described in section 3.3) in an attempt to improve population diversity (all prior examples were run with linear fitness scaling). Different sharing functions were

tested to determine which function shapes performed best for this particular design problem.

First, a set of runs was done with a linear sharing function ($\alpha = 1.0$) and $\sigma_{share} = 1.0$ (chromosomes share their fitness with all other chromosomes). Table 7.3 shows a comparison of these results with those results obtained without fitness sharing. In all cases, a steady-state GA was used with 17% overlap and the fitness function of (7a-d). Although this implementation of fitness sharing resulted in a slight decrease in the quality of the best individual found, it did result in a significant increase in population diversity, as shown in figure 7.5, without a significant increase in runtime. Unfortunately, much of these diversity came from the existence of poor designs (*i.e.*, designs not meeting the displacement constraint) in the final population.

	Best Fitness Score	Mass	Population Diversity	Runtime
No Sharing, popsize=30	1061	39%	11%	15±3
No Sharing, popsize=60	1063	37%	15%	30±5
Linear Sharing, popsize=30	1043	57%	44%	15±3
Linear Sharing, popsize=60	1054	46%	46%	32±5

Table 7.3 GA's with linear fitness sharing

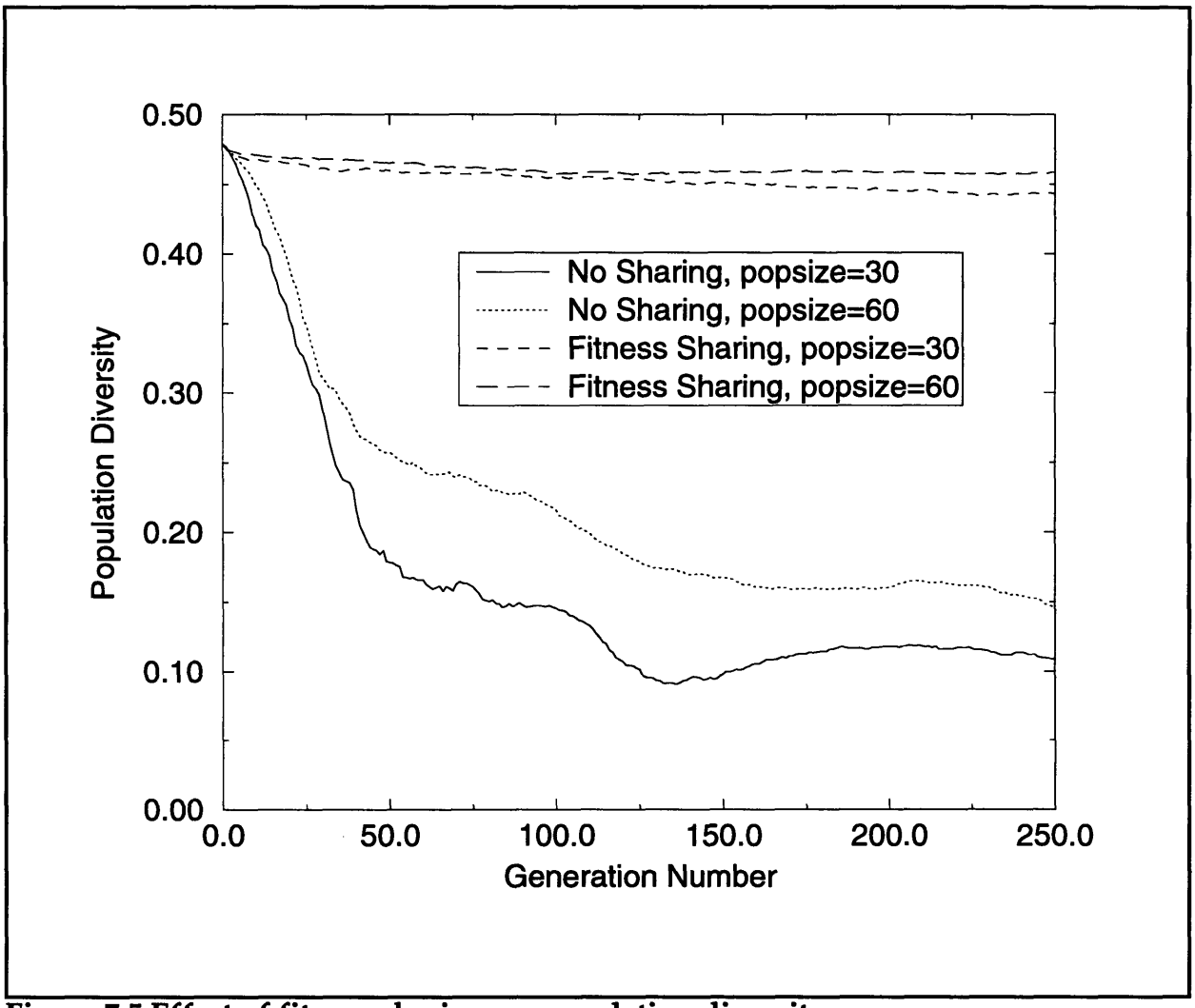


Figure 7.5 Effect of fitness sharing on population diversity

Next, different values of α (different shaped sharing functions) tested, with the results summarized in table 7.4. A population size of 60 was used in each case. It was found that a value of $\alpha = 2.0$ produced the best solutions and did so without decreasing population diversity. As a result, $\alpha = 2.0$ was used for all further runs. Runtimes were virtually identical (about 32 minutes / run ±5 minutes) regardless of the value of α.

	Best Fitness Score	Mass	Population Diversity
$\alpha = 0.5$	1055	45%	39%
$\alpha = 1.0$	1054	46%	40%
$\alpha = 2.0$	1057	43%	41%

Table 7.4 Fitness sharing with different values of α

7.2.4 Cluster Analysis

In the previous section it was found that speciating genetic algorithms (GA's with fitness sharing) were capable of producing final populations with increased levels of diversity. It is expected that members of the final population should be "clustered" about local optima. Cluster analysis, as described in section 6.3.3, was used to sort the final population of each GA run into discrete groups of similar structures, with a cutoff value of 0.85. Figure 7.6 shows the final population of a speciating GA run ($\alpha = 2.0$, population size = 60, 250 generations) divided into clusters, and table 7.5 shows the average and best fitness for each cluster.

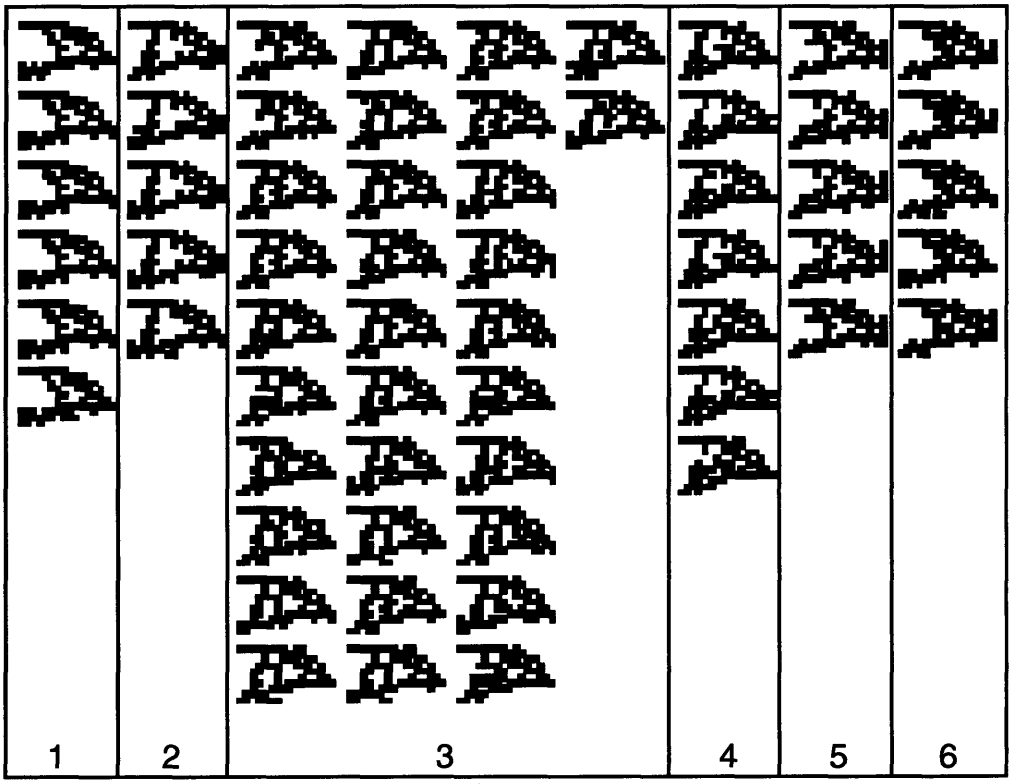


Figure 7.6 Clustered final population

Cluster	Best Fitness Score	Average Fitness Score
1	1062	1060
2	1060	1059
3	1060	1059
4	1059	1059
5	1059	1058
6	1059	1058

Table 7.5 Fitness of clusters

7.2.5 Restricted Mating

Once this cluster analysis was successfully implemented, it was used to restrict mating between dissimilar chromosomes. It has been found that, in general, poor children (also called "lethals") are produced by crossover between dissimilar chromosomes (Goldberg, 1989a, pgs. 192-197). In the following examples, the restricted mating does not begin until half way through the GA run (after 125 generations). At this point, mating is probabilistically restricted as described in section 6.3.3. Each generation, clustering is performed with a threshold value of 0.85. (*i.e.*, clusters less than 85% similar will not be combined). Table 7.6 shows the improved results obtained using this restricted mating technique. Both sets of runs were done with $\alpha = 2.0$, population size = 60, number of generations = 250.

Fitness Sharing	Restricted Mating	Best Fitness Score	Mass	Population Diversity	Number of Clusters	Runtime (minutes)
No	No	1063	37%	15%	2	30±3
Yes	No	1057	43%	41%	32	32±5
Yes	Yes	1061	39%	24%	14	45±6

Table 7.6 Summary of GA results (population size = 60)

7.3 Example 2: Beam Optimization

The next example considers a beam supported at each end and subjected to a vertical load at its midpoint. The design domain used was an 8x21 discretization, as shown in figure 7.8. The lower two nodes of each end element were constrained to have zero displacement, and the point load was applied to the node at the center of the element at the midpoint of the beam, shown in black in figure 7.7. The goal of this optimization was the same as in the previous example, to minimize the structure's weight while maintaining a displacement less than the specified value (δ_{max}) at the load point. The same fitness function (7a-d) was used. The magnitude of the applied load was set to 10 kN, and δ_{max} again was 0.01 m.

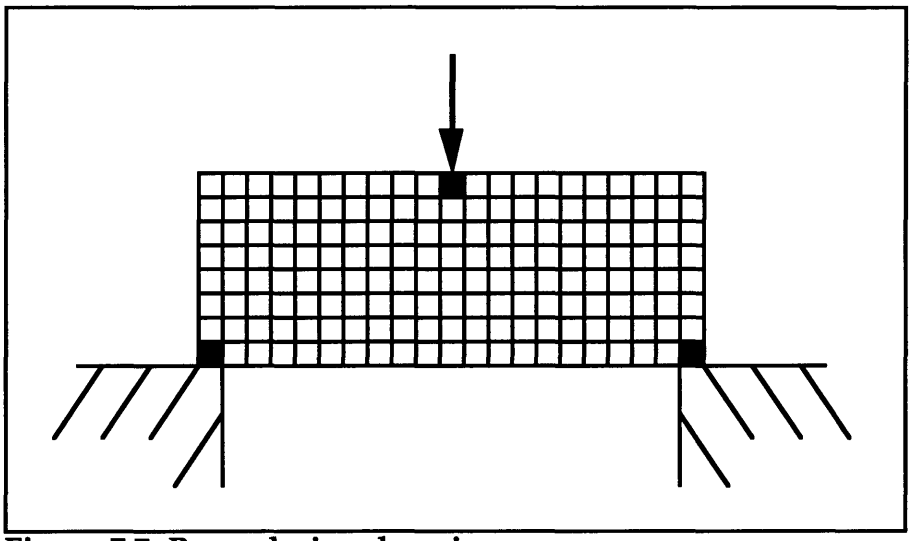


Figure 7.7 Beam design domain

As in the previous example, several GA runs were done with different settings. Runs were done with no fitness sharing or mating restrictions and with sharing but no mating restrictions. Runs with fitness sharing and restricted mating were also done, with populations of different sizes. In each case, the GA was run for 200 generations, and clustering was done with a cutoff value of 0.85. When restricted mating was implemented, it was started after 100 generations. Figures 7.8-7.10 show typical results from these GA runs. Table 7.7 summarizes the results of these GA runs. Note that each entry in this table represents the average result from multiple GA runs done with the same parameter settings.

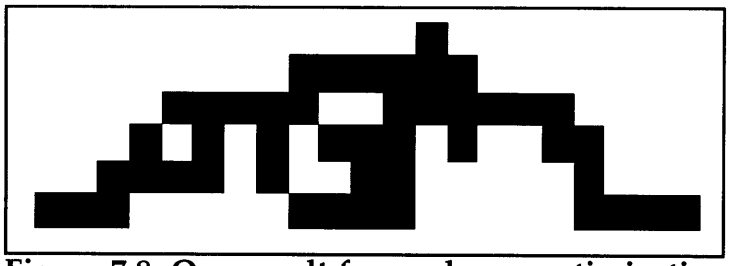


Figure 7.8 One result from a beam optimization run

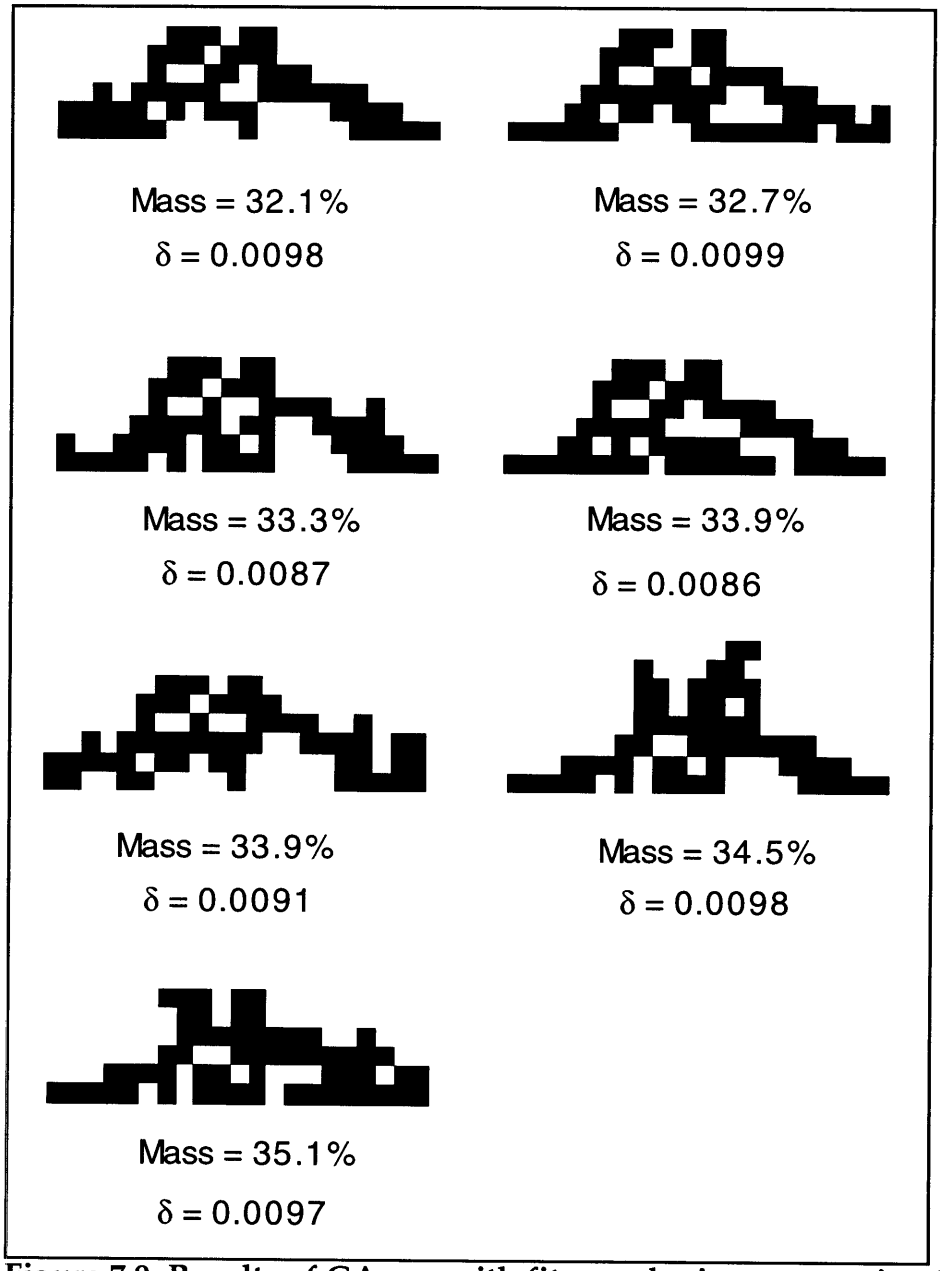


Figure 7.9 Results of GA run with fitness sharing, no restricted mating

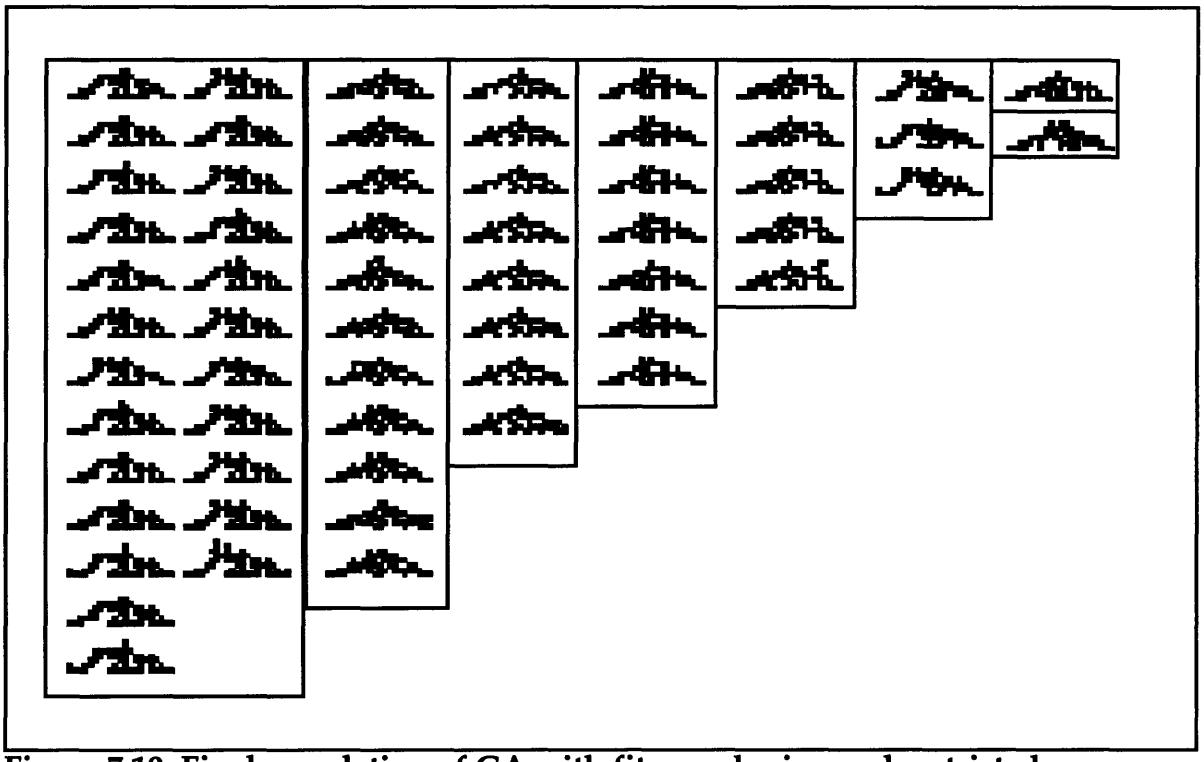


Figure 7.10 Final population of GA with fitness sharing and restricted mating. Individuals grouped by cluster.

	Population Size	Best Fitness Score	Mass	Number of Clusters
No fitness sharing	60	1072	28%	1
Fitness sharing, unrestricted mating	60	1069	31%	8
Fitness sharing, restricted mating	60	1071	29%	7
Fitness sharing, restricted mating	75	1071	29%	10
Fitness sharing, restricted mating	90	1071	29%	10

Table 7.7 Results of beam optimization

Chapter 8

Results:

Design of Adaptive Structures

8.1 Overview

This chapter presents examples of design optimization of adaptive structures performed using the hybrid genetic algorithm / simulated annealing method described in chapter 6. The effects of different neighborhood operators and fitness functions are examined.

8.2 Design Examples

As a basic example of this technique, the design of the cross-section of a beam is considered, as described in section 6.4, with the 10 x 10 design domain shown in figure 8.1. The element shown in black was used as the "seed" element. Material was required to be present there, and all material not connected to it by one or more edge-edge connections was removed via connectivity analysis. In the following examples, only the mass properties of the cross-section are considered. As a result, no material properties needed to be specified and no finite element analyses needed to be performed. Each element of the design domain is defined to have a length and width of 1. The mass of a cross section is defined as the number of material elements present after connectivity analysis, and the other mass properties (I_x, I_y, *etc.*) can easily be obtained by summing the contributions of each material element.

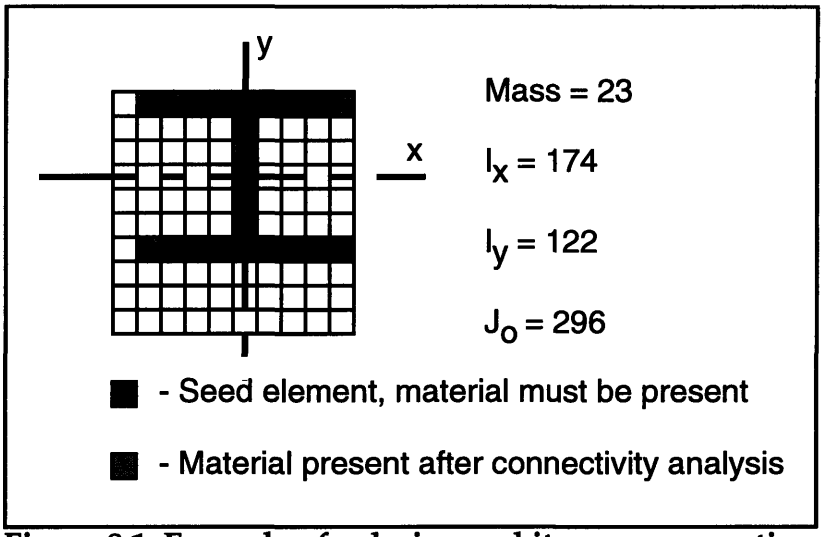


Figure 8.1 Example of a design and its mass properties

8.2.1 Neighborhood Operators

As discussed in section 6.4.2, the simulated annealing process requires the definition of a neighborhood operator to modify an existing topology. The choice of a neighborhood operator can have a great impact on the type of results produced by the simulated annealing optimization, as the operator controls how the structure adapts. In this investigation, a number of different neighborhood operators were implemented and tested. The most successful of these operators can be divided into two categories: move operators and flip operators.

Flip Operators

Flip operators work in much the same way as genetic algorithm mutation. For instance, one possible flip operator would be to randomly select one material element and turn it into a void element. An alternative flip operator would be to randomly select two elements, one material and one void, and to swap them (the material element becomes void and vice-versa).

Move Operators

Move operators are basically neighborhood operators which function by randomly choosing a material element in the design domain and moving it one unit (see figure 6.7a). Various move operators could allow material to move diagonally, or just horizontally or vertically. These move operators were thought to be more similar to how an actual structure might adapt.

Unfortunately, both types of operators have a common difficulty. Because connectivity analysis is performed after each annealing step, it is possible for the mass of a structure to change as it adapts. For example, if a move operator is used, a material element might move away from the element or elements connecting it to the "seed" element. If it does not make a new edge-edge connection in its new location, it will be removed when connectivity analysis is performed.

A number of variations on the operators described above were tested, and it was found that a one-point flip operator was the most effective at producing structures with high fitness scores. This operator worked as follows. At each annealing step, one element was randomly selected from the connected structure and its value was flipped (material to void or vice-versa). Connectivity analysis was then performed again. The result is that in any given step, the mass of the structure may increase or decrease, but, when a large number of steps are performed, the mass of the structure tends to remain approximately constant. All examples that follow use this one-point flip operator to perform the simulated annealing optimization.

8.2.2 Optimization Results

The goal of this example will be to generate a cross-section topology that has a high stiffness-to-weight ratio for bending about the x-axis (high I_x / mass) and can adapt to have a high stiffness-to-weight ratio for a torsional load applied about the cross-section's center of mass (high J_o / mass). As a first attempt, a fitness function was set up to evaluate a design as follows. First, connectivity analysis was performed. Next, the structure was analyzed to compute the stiffness-to-weight ratio for bending loads (I_x / mass). Simulated annealing was then performed to make the structure "adapt" to improve its performance under a torsional load (as measured by the ratio J_o / mass). The performance of the adapted structure was combined with the performance of the original structure as shown in equation 8.1 below.

$$Fitness = \left(\frac{I_x}{mass}\right)_{original\ structure} + \left(\frac{J_o}{mass}\right)_{Adapted\ structure} \tag{8.1}$$

Runs were done with the following parameters:

GA Type:	Steady-State GA
Pcross:	0.95

Pmut:	0.01
Population size:	75
Number of annealing steps:	500

The results of one run are shown in figure 8.2. The structure on the left was generated by the GA (it is the original structure), the structure on the right (the adapted structure) is the result of the annealing process. Table 8.1 gives the mass properties of the original and the adapted structures. Note that the annealing process did improve performance in response to torsional loads, although it significantly changed the topology to do so.

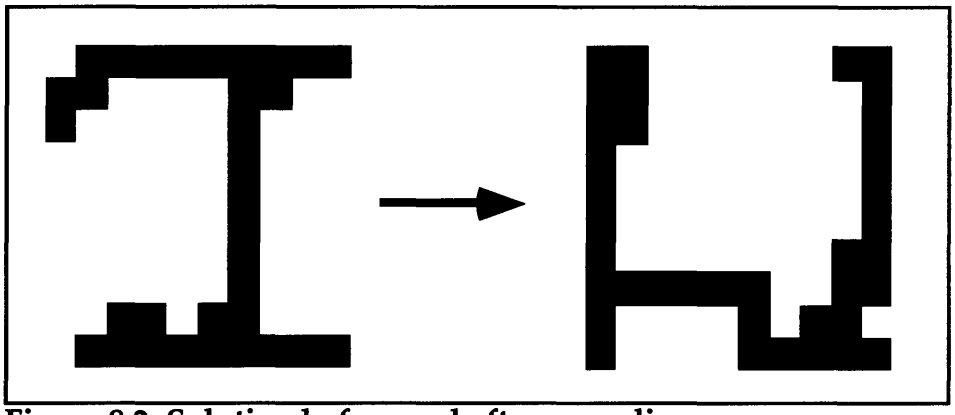


Figure 8.2 Solution before and after annealing

	I_x / mass	J_o / mass
original structure	14.8	21.2
adapted structure	9.6	23.7

Table 8.1 Properties of original and adapted structures

From looking at the design domain and the structures shown in figure 8.2, it is apparent that a C-shaped cross section could perform well with regards to both fitness criteria if it was oriented appropriately. If it is possible for a single topology to perform well under *both* sets of loading conditions (*i.e.*, either no or very little adaptation is required), then it is desired that the optimization process find such a structure. However, the fitness function above (eqn. 8.1) does not take into account the amount of adaptation performed, and is therefore unlikely to find such a structure. To account for this, a new fitness function was created, as shown in (8.2) below.

$$Fitness = (Fitness)_{original\ structure_{load\ 1}} + (Fitness)_{adapted\ structure_{load\ 2}} - Penalty \qquad (8.2)$$

where

$$Fitness_{original\ structure_{load\ 1}} = \frac{I_x}{mass} \tag{8.3}$$

$$Fitness_{Adapted\ structure_{load\ 2}} = \begin{cases} \frac{J_o}{mass}: & J_o < J_{\min} \\ \frac{J_{\min}}{mass}: & J_o \geq J_{\min} \end{cases} \tag{8.4}$$

$$Penalty = 20.0 * \left(\frac{\text{Number of annealing steps}}{500} \right) \tag{8.5}$$

With this fitness function, the annealing process stops as soon as it generates a structure with $J_o \geq J_{min}$ and the number of annealing steps taken to create this structure is recorded. If no structure is found that satisfies this constraint, the annealing process stops after 500 steps. The weighting factor of 20.0 was selected for the penalty function to give it approximately the same weight as the other two terms in the fitness function. A typical result from a run done with this fitness function and a constraint of $J_{min} = 880$ is shown in figure 8.3. This structure has the following mass properties:

mass =	35
I_x =	519.5
I_x / mass =	14.8
J_o =	880.7
J_o / mass =	25.2

Because $J_o > J_{min}$, no annealing needed to be done.

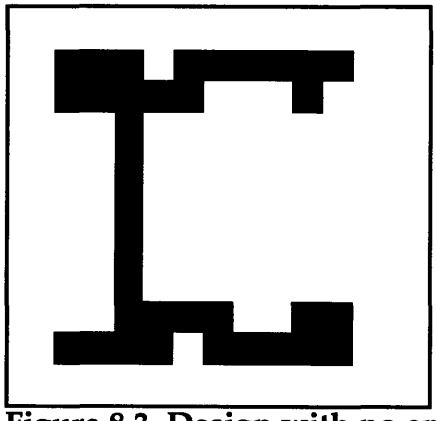


Figure 8.3 Design with no annealing required

Next, the constraint on the polar moment of inertia (J_{min}) was increased so that it would be less likely that a single design could have high performance under both loading conditions.

A run done with J_{min} = 1280 produced the results shown in figure 8.4. The mass properties of each structure are listed in table 8.2

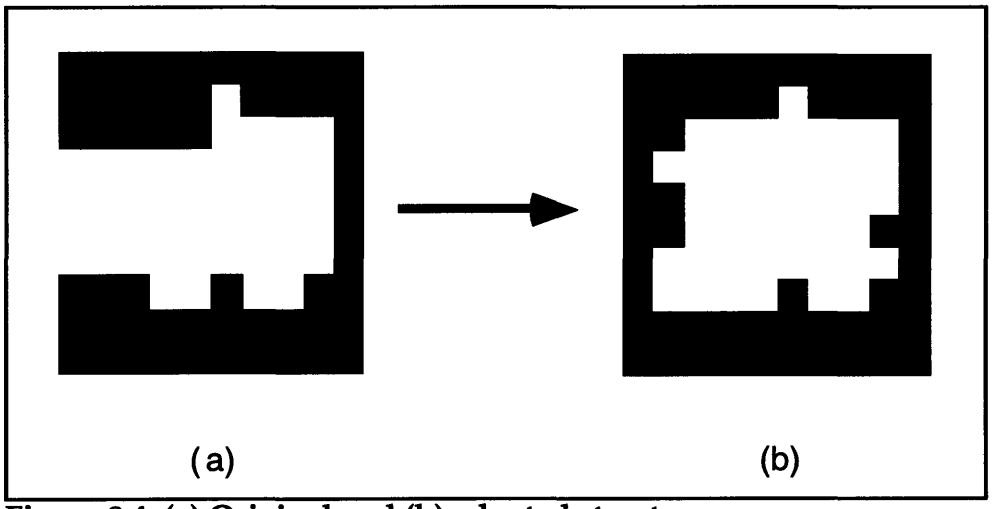


Figure 8.4 (a) Original and (b) adapted structures

	mass	I_x / mass	J_o	J_o / mass
original structure	55	13.1	1265	23.0
adapted structure	57	12.2	1335	23.4

Table 8.2 Properties of original and adapted structures

Chapter 9

Conclusions

9.1 Overview

This chapter reviews the contributions of this investigation and discusses the advantages and limitations of the optimization techniques presented. Finally, suggestions are made for potential areas of future work in the use of genetic algorithms for structural topology optimization.

9.2 Contributions of This Investigation

In this investigation, genetic algorithms were used to search for optimal structural topologies. More specifically, GA's were used to optimize the distribution of material in a discretized domain. Significant changes were made to the basic genetic algorithm approach to reduce run times and to increase diversity of solutions found. A hybrid genetic algorithm / simulated annealing approach was proposed for performing optimal design of adaptive structures. These optimization techniques were applied to example problems using a two-dimensional, binary (material / void) design domain.

9.2.1 Design of Planar Stress Structures

A genetic algorithm was used to perform structural topology optimization of 2-D structures under static loads. First, a steady-state genetic algorithm was implemented to increase the efficiency of this optimization process. This genetic algorithm used overlapping populations and was able to decrease the number of function evaluations required for an optimization run. This technique was applied to the optimization of a cantilevered plate, where the objective was to minimize the structure's weight subject to a constraint on the maximum allowable displacement at the point of load application.

Next, a new population initialization operator was created to further improve GA efficiency. This population initializer involved running a pre-processor GA in order to

create a population of "connected" structures (*i.e.*, structures in which the point of load application is connected to both constrained elements). This population of connected structures was then used as the initial population for the main GA run. The result was that the main GA did not have to spend several generations searching for feasible structures. Instead, it could begin the design optimization right away. Since the preprocessor GA performed no finite element analyses, it required very little time to run relative to the main GA. As a result, this new optimization technique of pre-processor GA + main GA was able to produce results superior to those obtained using traditional population initialization operators.

The effects of different types of connectivity analysis were then examined. Runs were done with no connectivity analysis, one-point connectivity (corner-corner connections allowed), and two-point connectivity (only edge-edge connections allowed). It was found that two-point connectivity was the most effective method for generating optimal topologies, producing slightly better results than either of the other two methods. A more drastic difference was found among the run-times required for these optimizations. Significantly less time was required for runs done with two-point connectivity. This was because finite element analysis is performed only on connected structures, and requiring two-point connectivity was the most effective method for avoiding the time-consuming analysis of poor designs.

It was also desired that a single GA run produce a final population containing multiple, different solutions. While the above modifications to the genetic algorithm were successful at improving efficiency and shortening run times, they did not result in any improvements in final population diversity. To achieve this, fitness sharing was implemented. Initial attempts at using fitness sharing resulted in a great increase in population diversity, but also led to a decrease in the quality of solutions produced. Although the final populations of such runs did contain a diverse set of solutions, many of these solutions were of significantly lower fitness than solutions found without fitness sharing.

Next, cluster analysis was implemented to divide the population of a GA run into different "species," where each species represented a group of similar designs. This analysis was then used to probabilistically restrict mating to similar chromosomes. It was found that restricting mating in this way helped to improve the quality of structure's found by reducing the number of "lethals" (poor designs resulting from the mating of dissimilar chromosomes) created by the GA. The best designs found using a GA with fitness sharing

and restricted mating were nearly as good as those produced by a GA without sharing or mating restrictions, and the diversity of solutions produced was increased significantly.

9.2.2 Design of Adaptive Structures

An optimization process involving the use of a genetic algorithm and simulated annealing was proposed for performing optimal topology design of adaptive structures. A sample problem involving the design of the cross-section of a beam subject to two sets of loading conditions was used to demonstrate the approach. Different types of simulated annealing neighborhood operators were implemented to examine their effectiveness.

Because initial results produced structures which had to adapt a great deal to improve performance under a changed loading condition, a new fitness function was implemented to reward structures requiring little adaptation to perform well under a changed load. With this new fitness function, the GA / simulated annealing approach was able to find a single structure capable of performing well under each set of loading conditions. When no such structure was possible, the optimization method produced a structure requiring only a relatively small amount of adaptation to account for the changed load.

9.3 Conclusions

The optimization methods detailed in this study were able to produce structural topologies of relatively high fitness. However, solutions generated using this methods would typically require some modifications before they could be considered truly optimal to actually manufacture and use. Topologies generated with these methods are frequently asymmetrical and often contain some "stray" material elements which could be removed with very little effect on the structure's performance. These shortcomings are a result of the probabilistic nature of genetic algorithms. Although GA's have trouble producing a truly optimal solution, they were found to perform well at locating nearly-optimal solutions in discrete, multi-modal design spaces.

9.3.1 Design of Planar Stress Structures

A difficulty encountered in using genetic algorithms for the design of planar stress structures was the computational cost required to perform the numerous finite element analyses required. Although this investigation implemented a number of methods for

improving GA efficiency, optimization was still computationally expensive. This computational cost limits the amount of discritization that may be used to represent a design. Using a design domain discretized as finely as those used in homogenization-based methods would still be prohibitively expensive in terms of required computation time.

In terms of maintaining a diverse population of individuals, the approach taken in this investigation was reasonably successful. In most cases, the GA was able to produce a final population containing a number of different optimal or nearly-optimal solutions. The effectiveness of this method was dependent on the fitness function used, however. Runs done with an unconstrained fitness function such as maximizing the stiffness-to-weight ratio of the structure still tended to converge to a single solution, although this convergence was slowed significantly.

Despite these shortcomings, the genetic algorithm approach performed reasonably well. Runtimes were reduced significantly from previous studies, and more diverse sets of solutions were generated. Such an approach also maintains the basic advantages of other genetic algorithm approaches, such as applicability to a wide range of problems and fitness functions, and the ability to perform well in an ill-behaved search space.

9.3.2 Design of Adaptive Structures

A hybrid genetic algorithm / simulated annealing method was introduced for the topology design of adaptive structures. A very simple example problem was used to demonstrate this basic approach. However, in order to implement this approach for a more realistic design problem, a number of difficulties must be overcome. One difficulty is in creating a representation which closely models an actual technique used for the design of adaptive structures. The sample problems in this investigation were intended to be demonstrative, and did not represent feasible adaptive structures. In particular, new neighborhood operators would need to be defined so that they more accurately modeled the way in which the actuators in an adaptive structure work. Further ideas for improved neighborhood operators are given in section 9.4.2

It was also found that the performance of this optimization technique was strongly dependent on the type of fitness function used. Optimizing the design of an adaptive structure involves multiple objective functions, and these objective functions must be combined to give one overall fitness score to each design, adding a further degree of

complexity to the problem. In this study, these multiple objective function scores were simply added together, with each term given approximately the same weight.

9.4 Future Work

Although the optimization techniques presented in this investigation performed reasonably well, there is still a great deal of room for improvement. The modifications to the basic GA approach succeeded in reducing runtimes, but computational expense is still a serious drawback to genetic algorithm based optimization methods when evaluating a design is computationally expensive. Investigations into approximate methods of evaluation could lead to shorter run-times, as could parallel evaluation of designs (*i.e.*, several processors could be used to simultaneously evaluate the members of a population). Because the computation required for the GA itself (selection, crossover, mutation, *etc.*), reducing the time required for evaluating individuals could drastically reduce overall run-times. Other modifications could be made to possibly improve genetic algorithm performance in terms of producing a diverse set of solutions. A variety of new methods for maintaining diversity in genetic algorithms are currently being developed. These new niching and speciation techniques could be implemented and compared in terms of quality and diversity of solutions produced to results obtained with fitness sharing.

In both plane stress and adaptive structure optimization problems, it was observed that the formulation of the fitness function had a great impact on the type and diversity of solutions produced. Further investigation could be made into the effects of different types of fitness functions, including the effects of constraints and penalties on GA performance. In the case of multi-objective optimization, more sophisticated methods for weighting and combining the objective scores could be implemented.

Several modifications could be made to the hybrid genetic algorithm / simulated annealing approach to allow it to approach a more realistic set of design problems. As described above, the particular fitness function used was found to have a great impact on the type of solutions generated with this method. New neighborhood operators could be created to more accurately model a structure's ability to adapt. Also, a limit could be placed on which parts of the topology are allowed to adapt. Adaptive structures use a finite number of actuators, and each actuator is capable of affecting only certain parts of the structure. By

incorporating this information into the annealing process, a more realistic "adapting" process could be defined and more realistic design problems could be attempted.

Computation time was not a significant issue in the adaptive structure design problems in this investigation as no finite element analyses were performed. However, if finite element or other computationally expensive analyses were required, the total time required for this optimization process would likely be exceedingly long. This is because each individual is analyzed once before the annealing process begins, and then several more times as the annealing process continues. Again, any methods capable of speeding up this structural analysis, such as using some sort of approximate analysis, would be a great help in reducing run-times.

References

Anagnostou, G., Ronquist, E., Patera, A., 1992, "A Computational Procedure for Part Design," *Computer Methods in Applied Mechanics and Engineering 97*, pps. 33-48.

Arora, J. S., 1989, *Introduction to Optimum Design,* McGraw-Hill, New York.

Ashley, S., 1995, "Smart Skis and other Adaptive Structures," *Mechanical Engineering*, Volume 117, pgs 76-81.

Baker, J., 1987, "Reducing Bias and Inefficiency in the Selection Algorithm," *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, Massachusetts Institute of Technology, July, pp. 14-21.

Bendsøe, M., Diaz, A., Kikuchi, N., 1993, "Topology and Generalized Layout Optimization of Elastic Structures," *Topology Design of Structures*, Bendsøe, M., Soares, C., eds., NATO ASI Series, pps. 159-205.

Bendsøe, M., Kikuchi, N., 1988, "Generating Optimal Topologies in Structural Design Using a Homogenization Method," *Computer Methods in Applied Mechanics and Engineering 71*, pps. 197-224.

Berke, L., Venkayya, B., 1974, "Review of Optimality Criteria Approaches to Structural Optimization," *ASME Winter Annual Meeting*, New York, November.

Chapman, C., 1994, *Structural Topology Optimization via the Genetic Algorithm*, Master of Science Thesis, Massachusettes Institute of Technology, May.

Chapman, C., Jakiela, M., 1996, "Genetic Algorithm-Based Structual Topology Design with Compliance and Topology Simplification Considerations," *ASME Journal of Mechanical Design,* Volume 118, Number 1, pgs 89-94.

Chapman, C., Saitou, K., Jakiela, M., 1994, "Genetic Algorithms as an Approach to Configuration and Topology Design," *ASME Journal of Mechanical Design*, Volume 116, pps. 1005-1012.

Chapman, C., Saitou, K., Jakiela, M., 1993, "Genetic Algorithms as an Approach to Configuration and Topology Design," *Proceedings of the ASME 19th Design Automation Conference: Advances in Design Automation*, Volume 1, American Society of Mechanical Engineers, DE-Volume 65-1, New York, pps. 485-498.

Cox, H., 1965, *The Design of Structures of Least Weight*, Pergamon Press, Oxford.

Davis, L. (ed.), 1991, *Handbook of Genetic Algorithms,* Van Nostrand Reinhold, New York.

Deb, K., Goldberg, D., 1989, "An Investigation of Niche and Species Formation in Genetic Function Optimization," *Proceedings of the Third International Conference on Genetic Algorithms,* Morgan Kaufmann Publishers, Inc., San Mateo California, pps. 42-50.

DeJong, K., 1975, "An Analysis of the Behavior of a Class of Genetic Adaptive Systems," Doctoral Dissertation, Department of Computer and Communication Sciences, The University of Michigan.

Eshelman, L. (ed.), 1995, Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufmann Publishers Inc., San Francisco.

Farin, G., 1993. *Curves and Surfaces for Computer Aided Geometric Design*, Academic Press, San Diego.

Gallagher, R., 1973, "Fully Stressed Design," in R.H. Gallagher and O.C. Zienkiewicz (eds.), *Optimum Structural Design*, John Wiley and Sons.

Goldberg, D., 1989a, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Massachusetts.

Goldberg, D., 1989b, "Sizing Populations for Serial and Parallel Genetic Algorithms," *Proceedings of the Third International Conference on Genetic Algorithms*, Schaffer, J., ed., George Mason University, pps. 70-79.

Goldberg, D., Richardson, J., 1987, "Genetic Algorithms with Sharing for Multimodal Function Optimization," *Proceedings of the Second International Conference on Genetic Algorithms,* Lawrence Earlbaum Associates, Hillsdale New Jersey, pps. 41-49,

Goldberg, D., Samtani, M., 1986, "Engineering Optimization via Genetic Algorithm," *Electronic Computation--Proceedings of the Ninth Conference on Electronic Computation*, Published by the American Society of Civil Engineers, University of Alabama at Birmingham, February, pps. 471-482.

Grefenstette, J., 1986, "Optimization of Control Parameters for Genetic Algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, Volume SMC-16, pps. 122-128.

Hajela, P., 1990, "Genetic Search-An Approach to the Nonconvex Optimization Problem," *AIAA Journal*, Volume 28, Number 7, pps. 1205-1210.

Hemp, W., 1973, *Optimum Structures*, Oxford University Press, London.

Holland, J., 1975, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, Michigan.

Jenkins, W., 1991a, "Structural Optimisation with the Genetic Algorithm," *The Structural Engineer,* Volume 69, Number 24, pps. 418-422.

Jenkins, W., 1991b, "Towards Structural Optimisation via the Genetic Algorithm," *Computers & Structures*, Volume 40, Number 5, pps. 1321-1327.

Jensen, E., 1992, *Topological Structural Design Using Genetic Algorithms*, Doctor of Philosophy Thesis, Purdue University, November.

Kane, C., Jouve, F., Schoenauer, M., "Structural Topology Optimization in Linear and Nonlinear Elasticity Using Genetic Algorithms," *ASME Design Engineering Technical Conferences*, Volume DE-82, pps. 385-392.

Kirkpatrick, S., Gelatt, C., Vecchi, M., 1983, "Optimization by Simulated Annealing," *Science*, Volume 220, pps. 671-680.

Kirsch, U., 1993, *Structural Optimization: Fundamentals and Applications,* Springer-Verlag, Berlin, Heidelberg, New York.

Kirsch, U., 1981, *Optimum Structural Design*, McGaw-Hill Book Company, New York.

Kohn, R., Strang, G., 1986, "Optimal Design and Relaxationof Variational Problems, I, II, III," *Communications on Pure and Applied Mathematics*, Volume 39, pps. 113-137, 139-182, 353-377.

Koza, J., 1992, *Genetic Programming: On the Programming of Computers by Means of Natural Selection,* MIT Press, Cambridge, Massachusetts.

Lev, O., 1981, *Structural Optimization: Recent Developments and Applications*, , Published by the American Society of Civil Engineers, New York.

Mahfoud, S., 1995, *Niching Methods for Genetic Algorithms*, Doctor of Philosophy Thesis, University of Illinois at Urbana-Champaign.

Michalewicz, 1994, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag,Berlin.

Mitchell, A., 1904, "The Limits of Economy of Material in Frame Structures," *Philosophical Magazine*, Series 6, Volume 8, pps. 589-597.

Papalambros, P., Chirehdast, M., 1990, "An Integrated Environment for Structural Configuration Design," *Journal of Engineering Design*, Volume 1, Number 1, pps. 73-96.

Rajeev, S., Krishnamoorthy, C., 1992, "Discrete Optimization of Structures Using Genetic Algorithms," *Journal of Structural Engineering*, Volume 118, Number 5, pps. 1233-1250.

Reddy, G., Cagan, J., 1995, "An Improved Shape Annealing Algorithm for Truss Topology Generation," *ASME Journal of Mechanical Design*, Volume 117, No. 2(A), pps. 315-321.

Richards, R., Sheppard, S., 1992, "Learning Classifier Systems in Design Optimization," *Proceedings of the 1992 Design Theory and Methodology Conference*, Taylor, D., Stauffer, L., eds., DE-Volume 42, Published by the American Society of Mechanical Engineers, Scottsdale, Arizona, pps. 179-186.

Romesburg, C., 1984, *Cluster Analysis for Researchers,* Lifetime Learning Publications (Wadsworth), Belmont California.

Sandgren, E., Jensen, E., 1992, "Automotive Structural Design Employing a Genetic Optimization Algorithm," *SAE Technical Paper #920772*, Proceedings of the SAE International Congress and Exposition, Detroit, February.

Sandgren, E., Jensen, E., Welton, J., 1990, "Topological Design of Structural Components Using Genetic Optimization Methods," *Sensitivity Analysis and Optimization with Numerical Methods*, AMD-Vol. 115, Proceedings of the Winter Annual Meeting of the American Society of Mechanical Engineers, Dallas, Texas, pps. 31-43.

Schaffer, J., Caruana, R., Eshelman, L., and Das, R., 1989, "A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization," *Proceedings of the Third International Conference on Genetic Algorithms*, George Mason University, June, pp. 51-60.

Strang, G., Kohn, R., 1986, "Optimal Design in Elasticity and Plasticity," *International Journal for Numerical Methods in Engineering*, Volume 22, pps. 183-188.

Suzuki, K., Kikuchi, N., 1991, "A Homogenization Method for Shape and Topology Optimization," *Computer Methods in Applied Mechanics and Engineering*, Volume 93, pps. 291-318.

Syswerda, G., 1989, "Uniform Crossover in Genetic Algorithms," *Proceedings of the Third International Conference on Genetic Algorithms*, George Mason University, June, pp. 2-9.

Syswerda, G., 1991, "A Study of Reproduction in Generational and Steady-State Genetic Algorithms," in G. J. E. Rawlins (ed.), *Foundations of Genetic Algorithms* , Morgan Kaufmann, pps. 94-101.

Watabe, H., and Okino, N., 1993, "A Study on Genetic Shape Design," *Proceedings of the Fifth International Conference on Genetic Algorithms*, University of Illinois at Urbana-Champaign, July, pp. 445-450.