Automated Generation and Analysis of Dynamic System Designs

by

Eng Hock (Francis) Tay

B.Eng., National University of Singapore
1986
M.Eng., National University of Singapore
1991

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

at the

Massachusetts Institute of Technology
October 1995

Signature of Author _____
Department of Mechanical Engineering
October 19, 1995

Certified by _____
Professor Woodie C. Flowers
Thesis Supervisor

Accepted by _____
Professor Ain A. Sonin
Chairman, Department Committee

Automated Generation and Analysis of Dynamic System Designs

by

Eng Hock Tay

Submitted to the Department of Mechanical Engineering
on 19 October 1995 in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy

Abstract

This thesis uses genetic algorithms (GAs) to suggest new dynamic systems based on topological remapping of system constituents. The bond graph representation of the dynamic system behavior is evolved by the operators encapsulated in the genetic algorithms to meet the specified design criteria. The resultant evolved graph is assembled by designers with schemes to produce design variants.

Behavioral transformation and structural transformation are adopted as strategies to generate design variants that extend beyond the scope of parametric design into innovative design. Behavioral transformation involves changes in the structure of the representation graphs, while maintaining the functions. Structural transformation involves changes in the components and the subsystems represented by the graph fragments.

This research is an investigation into how design operators such as mutation, combination and selection can be used to transform computationally a representation of one design to another. This forms the central part of a long-term objective of developing a computational environment to facilitate the entire design process as much as possible. The benefits of design variants generation include an extensive and unbiased search of the problem-solution space leading to unanticipated designs, and increased efficiency in searching. Moreover, the second stage of the developed computational platform allows the optimization of the generated variants.

GAs are used to implement the operators of the transformation to search the problem-solution space because GAs are very robust search routines. Further, since the goal is to generate many solutions, genetic speciation is used to diverge the search so as to uncover other desirable solutions. However, GAs are knowledge-lean. To overcome this shortcoming, bond graphs are used to represent the domain knowledge. Bond graphs provide a unified approach to the analysis, synthesis and evaluation of dynamic engineering

systems. Though the scope of this investigation is limited to systems represented by bond graphs, the domain is wide enough to include many interesting applications like pump systems and vibration isolation systems.

Thesis Committee:
Professor Woodie C. Flowers, Chairman
Professor David Gordon Wilson
Assistant Professor Anna Thornton
Dr John Barrus

To my beloved wife, Christine.

Acknowledgements

My heartfelt gratitude goes to my supervisor, Professor Woodie Flowers, for his unfailing support and wise counsel. From Professor Flowers, I have learned that an individual can make a difference. He has shown tremendous drive in his contribution to education in general, and to design education in particular. I wish to thank my committee members, Professors Dave Wilson and Anna Thornton, and Dr John Barrus, for their advice and guidance. I am deeply indebted to Dr John Barrus for his computing assistance. John's patience and helpfulness really set a new standard for me to emulate. I am grateful to Matthew Wall who has written the genetic algorithms library which was utilized heavily to develop this work. I am thankful to Professor Alex Slocum for his advice in the early part of this work. I am also grateful to Professor Nam Suh for the year of guidance in research that he gave me.

Sincere appreciation is expressed to the National University of Singapore (NUS) for the financial support during my graduate studies at Massachusetts Institute of Technology (MIT). I wish to thank all my lecturers and professors at NUS, who have taught me and thereby started me on this quest for knowledge. In particular, I wish to thank Associate Professor Poo Aun Ngiauw and Dr Lim Kah Bin for helping me to become a faculty member in the University, Professor Andrew Nee and Dr Wong Yoke San for guiding me through my first major research endeavour, the Master of Engineering degree, and Associate Professors Chew Yong Tian and Lim Siak Pin for their fine examples as teachers.

Many thanks to my office mates in Room 3-443 who have made my stay at MIT more interesting. I am certainly privileged to be able to associate with these highly talented people. Thank you, Benjamin Linder, Matthew Wall, Li Shu and Julie Yang.

I wish to express my deepest thanks to some very special people in my life: my beloved wife, Christine, thank you for making me whole, for being there when I needed someone, my deepest gratitude for your understanding, patience and love; my daughter and son, Charisse and Benjamin, thank you for the smiles and laughter, reassuring me that it was all worth the effort; my mother, Chia Boh and my late father, Koh Tiong for their love and upbringing ; my father-in-law and my mother-in-law for their support and my brothers and sisters for their understanding.

Finally, I wish to thank God for His divine guidance and mercy, without which, this thesis would not be possible. Certainly the fear of the Lord is the beginning of wisdom.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.0 Overview

The focus of this thesis is computer-assisted conceptual design of dynamic systems. A model for computationally generating conceptual design variants for a class of applications is proposed. The model is tested via a computer program, and several case studies are conducted. However, in order to appreciate and understand fully the implications of the case studies, a number of issues must be examined. First, the problem perspective is outlined. Next, the objectives of the research are clearly described. Since there are many different terminologies used by design researchers, an example is presented to illustrate the type of design this research has investigated. The scope of the research is then specified. A structure of the dissertation is included to aid the reader.

1.1 Problem Perspective

Generaing a conceptual design computationally is difficult because it requires an algorithm. The word, algorithm, is used here to mean a well-defined and formalized procedure. Hitherto, not much work has been done to formalize conceptual design.

Some research and development done on computer-aided design are not based on the design models and theories put forth by design researchers, owing to a number of factors including the following:

1. The current development of computational tools concentrates on a specfic stage of design, while most of the existing design models cover the entire design process. Although this may be acceptable in some cases, it may not be ideal. For instance, if an idea cannot be represented by the existing computational design tools, that idea cannot be conveyed in as much detail as ideas that *can* be represented by the design tools. Those ideas that are presented in great detail are more likely to be accepted, as they appear to be better developed than the rest. An example of this might be Velcro™. If the idea of some plant sticking to the socks of a designer is compared to a sketched drawing of a strap design, the strap design might be accepted, as it gives the impression of being better developed.

2. Some of the design models put forward to represent the design process lack the structure and formality needed to automate them.

3. The computational tools for design have traditionally concentrated on the detail-drawing stage, although there have been shifts in recent years. ProEngineer™ and research [Ramaswamy 93][Slocum 92] in the use of spreadsheets as an engineering design tool can track the design process at an earlier stage. However, these tools have yet to find their applications in dynamic systems, that is, systems that have dynamic characteristics such as stability and phase margin.

This research addresses the issues outlined above. Specifically, the research focuses on the generation of design variants of a dynamic system, given the desired dynamic characteristics. The results of the program are in the form of bond graph structures which can be conveniently converted to schematic drawings for the initiated. From the schematic drawing, detailed drawings can then be generated. All the outputs of

the program have the same degree of detail. Such outputs are unbiased representations of the options available.

1.2 Research Objectives and Contributions

The primary contribution of this research is to provide a foundation for a computational model of the conceptual design of dynamic systems. A computational model has three main characteristics: design representation, operators of transformation, and an objective function expressing the design metric. The computational model of design will allow the automation of the process that it represents. The computational model requires the identification and incorporation of the design operators that will "move" the process within the design model. This process is modeled in terms of the design representation and the objective function against which the design representation is evaluated.

In this research, the objectives are to develop:

1. An understanding of the transformation that takes place in the conceptual design phase whereby design requirements are mapped onto a design description.
2. A computational model for the conceptual design of dynamic systems based on the understanding of the transformation.
3. A computer program to test some of the insights developed. The operators that perform the above-mentioned transformation have to be incorporated into the program. In a sense, the program has to become the embodiment of the design model being developed.

The long-term objective of the author is to develop a computational process to facilitate as much as possible the design process of dynamic systems. Such a computational system is intended to support the design process by helping the designer

explore a wider range of possible designs. This may be especially helpful to a designer who, because of time pressure, may not explore the design space as widely as possible, and therefore may limit his search to familiar solutions.

The findings and development of this research are:

1. The transformation that maps design requirements onto design descriptions of dynamic systems has been implemented using the genetic operators of combination, mutation and selection. Further, a genetic strategy to form groups of designs, called "speciation," has been applied successfully in the generation of design variants.

2. A design model that is appropriate for computerization, and that extends the problem-solution space, has been developed and tested to be effective in the design of dynamic systems. (This will be elaborated in Chapter 4.)

3. A computer program has been developed to embody the design model. In this program, a genetic algorithm is used to generate and transform design variants; MATLAB™ is incorporated to provide the spectrum of dynamic characteristics for evaluation; and bond graphs are used to provide the means of interface between the designer and the program.

To understand better this research, and the issues that it addresses, an illustrative example is presented next. (This example is taken from the case study found in Chapter 7 of this dissertation.)

1.3 An Illustrative Problem

Consider the design of the vibration isolation system shown in Figure 1.1. The goal of such a system is to isolate a mechanical system from the vibration of the foundation. This case is selected for illustration because lumped-parameter mathematical

models are frequently used in the analysis and design of vibration isolation systems, as well as in the interpretation of characteristics of vibrating mechanical systems. The bond graph representation used in this research is ideal for modeling lumped-parameter problems. Moreover, vibration isolation is an important engineering problem because it is found in many applications such as machine structure design, packaging, and vehicle suspension.

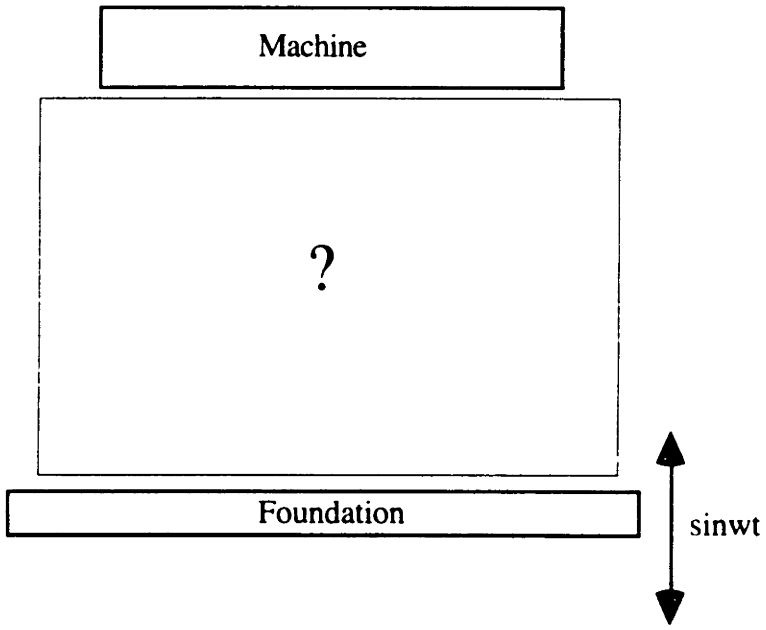


Figure 1.1 Vibration isolation problem

Better design typically results from generating many design alternatives. This will allow the designer to develop insights in the design, and thereby to arrive at better hybrid design [Pugh 90].

In practice, the configuration of the vibration isolation system is decided first, and then the dynamic characteristics are determined. In this research, the desired dynamic characteristics are formulated, and then the configuration of the system is designed to fulfill the requirements.

As shown in Figure 1.1, a machine is to be isolated from a vibrating foundation. The design requirements of the dynamic system are:

1. Absolute transmissibility (ratio of the velocity of the machine to the velocity of the foundation) < 0.0001.
2. Order of system (the number of state variables) must not be greater than four.

The outputs from the computer program are in the form of bond graphs, which can in turn be converted directly to the mechanical schematic of masses, springs and dash pots. (This is shown in Figure 1.2.) Moreover, the values of the parameters describing the components are determined.

The full details of the generation are described in Chapter 7. At this juncture, it must be emphasized that the design process has proceeded in the following manner:

1. Formulate the dynamic characteristics of the design.
2. Generate various representation (graphs) that will produce dynamic behavior.
3. Select some probable configurations.
4. Optimize the parameters of the desired dynamic design characteristics.
5. Embody the bond graph with a physical description.

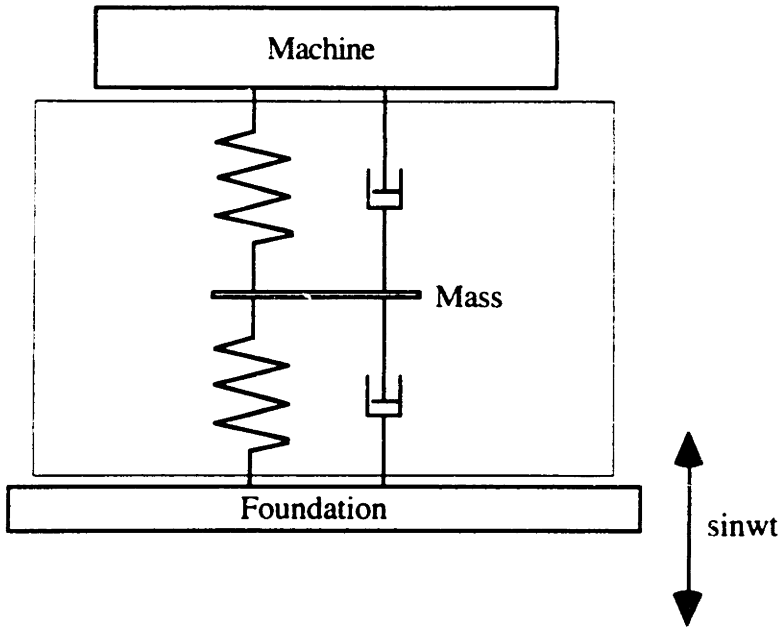


Figure 1.2 One of the solutions generated by the program

This proposed design process is to be contrasted with the traditional approach in the design of dynamic systems, which is as follows:

1. Formulate the dynamic design characteristics.
2. Select a configuration for the components.
3. Select values for the parameters associated with the components in that configuration.
4. Model the system.
5. Determine the dynamic characteristics and alter the values to satisfy the dynamic design characteristics.

The former method may not necessarily produce a good solution, but it does provide an opportunity to explore as wide a space of design solutions as possible.

With additional design information and specifications such as weight, cost and space, the design selection can be further narrowed. Since the thrust of this research is the automation of the conceptual design process of dynamic systems, the results are left as they were. At this stage it is important to note that the method used in generating the variants is very generic and robust. Therefore, it will not be difficult to incorporate further design information if desired.

1.4 Scope of Investigation

Bond graphs are used to represent the dynamic systems. Though bond graphs are versatile in representing inter-domain quantities, they have some restrictions. These restrictions confine the investigation to:

1. State determined systems.
2. Systems that can be described as networks of lumped-parameter, idealized elements in the translational-mechanical, rotational-mechanical, fluid-mechanical, and electrical media.

Such systems include pump systems, vibration isolation systems, accelerometers and pressure gauges.

1.5 Structure of this Dissertation

This thesis is divided into four (4) parts as shown in Figure 1.3 (page 9). The first part begins by describing the objectives of this research and proposing a solution. Background information necessary for a good understanding of this research is provided. The second part reviews the previous works on design models, bond graphs and genetic algorithms, and situates the research in the proper context. The third part outlines the

approach adopted in this research. The representation of the design model and the incorporation of the design operators are described. The results are shown. Some examples are included to illustrate the utility of the computational tool. The final part summarizes the contributions of this investigation and provides conclusions regarding the abilities and limitations of the operators-based computational design model.

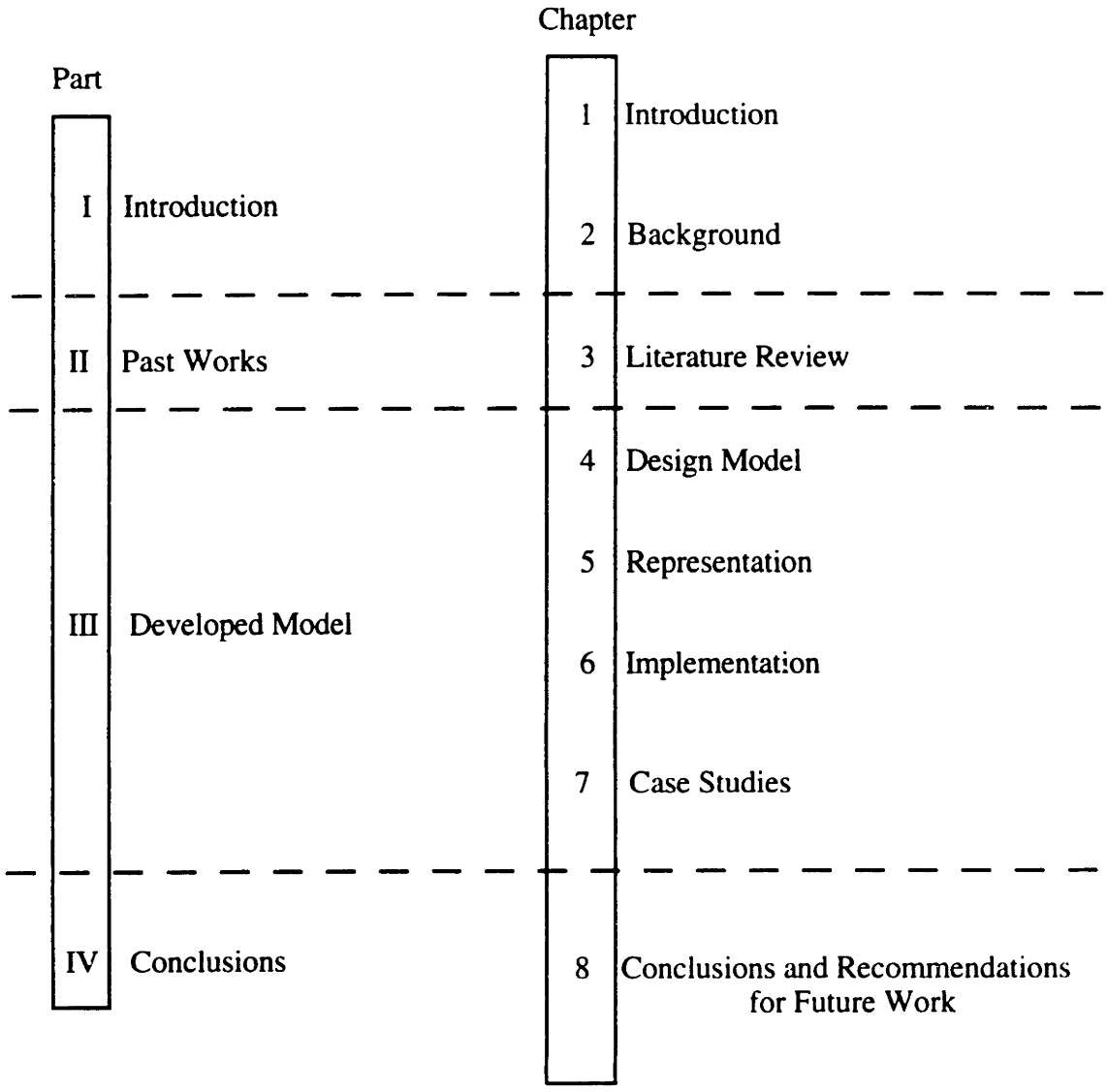


Figure 1.3 Organization of the dissertation

Chapter 2

Background

2.0 Overview

This chapter first introduces and defines the domain of this research, which is conceptual design. Next, a computational model for design is described to illustrate the essentials of such a model. The bond graph representation is then described. A justification of the chosen representation is included. The chapter concludes with a discussion of genetic algorithms which were used to embody the operators of the computational model. The discussion includes the intuition behind the genetic algorithm used in the design process.

2.1 Conceptual Design

In this dissertation, conceptual design is defined as that part of the design process that starts with a problem statement in the form of a set of functional requirements. The result of the conceptual design phase is a set of descriptions of designs proposed to meet the functional requirements.

Conceptual design is perhaps the single most important, and least understood, step in the product development process. Conceptual design is the driving factor in determining product quality and time-to-market, and it is estimated that 60% of all life-cycle costs are fixed during the conceptual design [Westinghouse 84]. Downstream processes such as detailed design, manufacture, and inspection cannot make up for poorly developed conceptual designs.

Conceptual design comprises many activities such as idea generation, decomposition, analysis, evaluation and synthesis. There are various models of design proposed by design researchers and there are major differences in the various design models. However, it is generally agreed that the generation of many design variants is the only way of assuring good design [West 92].

2.2 Design Variants

A variant, as defined in the *American Heritage Dictionary*, is something that differs in form only slightly from something else. In this thesis, "design variant" is used to describe designs that differ in the following ways:

1. Values of the parameters. The word "parameter" is used to mean a distinguishing characteristic or feature that defines a component. The parameter of the component determines its behavior, and is the quantity which is varied in an experiment. Some examples of parameters in dynamic systems are spring stiffness and flywheel mass. The values of the parameters may serve to indicate the type of physical element to be used. For example, compression springs may be used up to a certain point before they buckle.
2. The components. These can be removed, added or changed. For example, the lever with a certain mechanical advantage expressed in length ratio may be replaced by a hydraulic transformer with the mechanical advantage expressed in area ratio. When a

parameter such as stiffness is removed, it means the spring has been eliminated in the mechanical domain.

3. The configuration of the design. This involves the rearrangement of components, possibly leading to a new design.

4. A combination of the above. Though the above-mentioned individual change alters the original design by a "step," a combination of changes can drastically alter the design. These different designs are still design variants, as they can be traced to the original design in a step-by-step reformulation. In this sense, the design variants differ in form only "slightly" from each other.

2.3 Computational Model

The goal of developing a computational model is to automate the process for which the model is developed. Such automation is desired in design or part of design, as it enables the designers to concentrate their resources on other tasks.

Essential to the construction of a computational model are the representation and the process, corresponding to the data structure and the algorithm in computing. The process of the computational model can be further broken down to transformation operators and design metric. The operators transform the representation in accordance with the design metric.

Some researchers [Gero et al 90, 91] have classified design in computational terms:

1. Creative design can be defined as that design activity which occurs when a new variable is introduced into the design.

2. Innovative design can be defined as that design activity which occurs when the context which constrains the available ranges of the values for the variables is jettisoned so that unexpected values become possible.

3. Routine design can be defined as that design activity which occurs when all the necessary knowledge is available. It may be expressed as being that design activity which occurs when all the knowledge about the variables – objectives expressed in terms of these variables, constraints expressed in terms of these variables and the processes needed to find values for these variables – is known *a priori*.

Typical computational models of design can be grouped by the processes they utilize, including simulation, optimization, generation, decomposition, constraint satisfaction, and more generally, searching. All these processes share one concept, namely that structures are produced in a design process, and their resultant behaviors are evaluated.

In this thesis, search operators such as combination, selection and mutation are used to transform the bond graph representation. A strategy called "speciation" is used to generate different topologies of bond graphs. The bond graph representation is suitable for computer mechanization because bond graphs can be formally defined. (The ability of the representation to be formally defined is a necessary condition for it to be computer-mechanized.)

The design variants must be represented in such a way that the changes mentioned above can be performed. For this purpose, and more which will be elaborated in Section 2.4.4, bond graphs are used.

2.4 Bond Graph

2.4.1 Introduction

Bond graphs are concise graph structures that provide a unified approach to modeling and manipulating dynamic engineering systems. Since their inception by Paynter [Paynter 61], the number of published uses of bond graph techniques has increased exponentially [Beaman 88]. Bond graphs have been used extensively in many application areas, including vacuum cleaners [Remmerswaal 85], robotic manipulators [Margolis 79], and torque converters [Hrovat 85]. A standard description of the bond graph method of physical modeling can be found in the text by Rosenberg and Karnopp [Rosenberg 83].

One goal of modeling is to obtain a functional representation of a physical system. In this section, the use of bond graph representation is justified. A brief description of bond graphs is included for completeness. A more thorough treatment of bond graphs can be found in [Rosenberg 83].

2.4.2 Representation

Bond graphs are composed of a structure of nodes that are connected by bonds. The nodes of bond graphs are idealized elements which correspond to different physical properties, such as capacitances, inertias and transformers. (This is shown in Table 2.1.) The nodes do not correspond to components specifically, but rather to phenomena and principles.

There are three types of nodes: 1-port, 2-port and N-port. The designations indicate the number of bonds that are connected to each node. The behavior of the nodes is

described by constitutive equations, power flow, and causal relationships. The basic physics involved in the description of mechanical, electrical, hydraulic, thermal, magnetic, and fluid dynamic systems is phrased in terms of four types of generalized variables: effort, flow, displacement and momentum.

Table 2.1 Bond graph elements

Element	Domain			
	Mechanical Translational	Mechanical Rotational	Hydraulic	Electrical
I	Mass	Flywheel	Fluid Mass	Inductor
C	Spring	Torsional Spring	Fluid Volume	Capacitor
R	Damper	Rotational Damper	Orifice	Resistor

1-port nodes (or elements) dissipate power, store energy, and supply power. Dampers, springs and masses are the mechanical elements represented by the passive 1-port elements. Force (effort) and velocity (flow) sources are represented as active 1-port nodes.

2-port elements transform power. Transformers are 2-port elements that represent an imposed proportionate relationship between similar quantities; for example, a gear pair constrains rotational speeds. Gyrators are 2-port elements that impose a proportionate relationship between dual quantities; for example, a torque converter constrains the

relationship between torque and angular velocity. Power is conserved across a 2-port element.

N-port elements represent the structure of the system corresponding to the connections among the elemen . There are two types of N-port elements: 0-junctions and 1-junctions, which correspond respectively to "same force" and "same velocity" connections. Equivalents of Kirchoff's laws apply to N-port elements: the sum of the flows around a 0-junction is zero (the bonds share a common effort); the sum of the efforts around a 1-junction is zero (the bonds share a common flow).

The nodes are connected by bonds. The state of the system is represented in terms of generalized efforts and flows along the bonds. These generalized quantities correspond to different quantities in different physical domains. For example, the effort and flow correspondents in the translation domain are force and velocity. This choice of variables makes it possible to model systems consisting of components from different physical domains by means of a set of generalized elements. This is the characteristic that will be exploited in the future to implement analogy in the generation of design variants.

An example taken from [Martens 72] is used to demonstrate the unified nature of bond graphs. Figure 2.1 is a schematic view of an air pump. It is a vibratory pump in which an electromagnetic circuit drives a small permanent magnet attached to a pivotal lever, which in turn drives a rubber bellows pump. The bellows pump has rubber check valves and delivers a small flow of air.

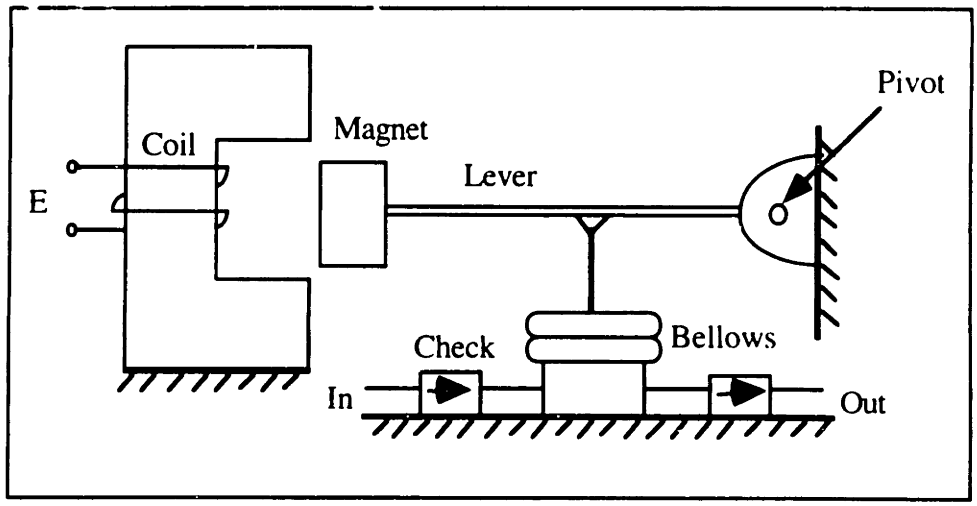


Figure 2.1 Schematic of air pump

The corresponding bond graph representation is as follows:

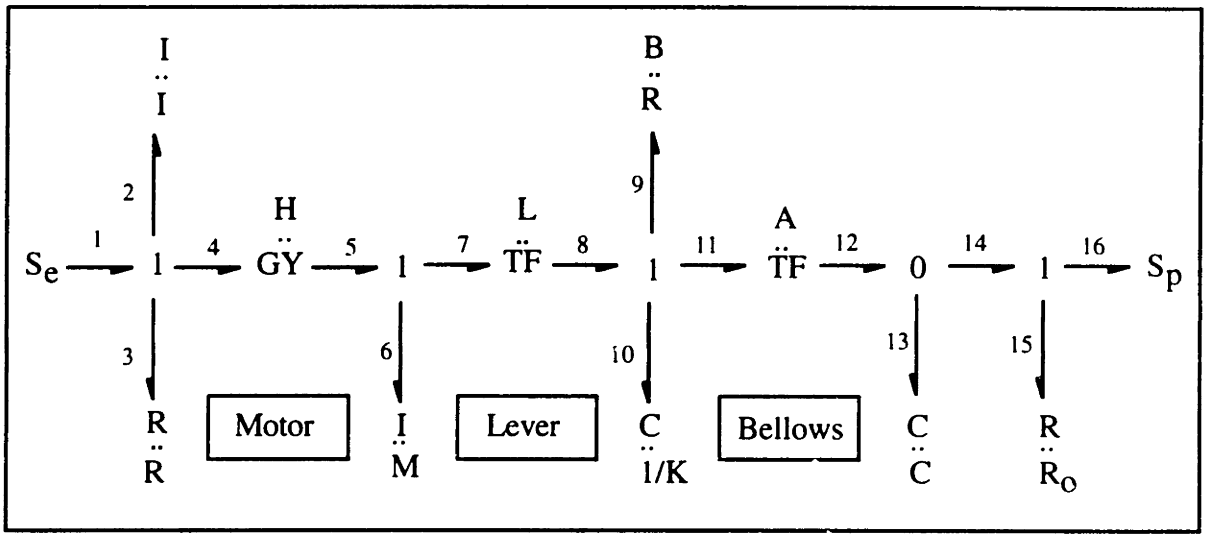


Figure 2.2 Bond graph for the air pump

Although there is a one-to-one correspondence between the physical components and the bond graph fragments (section of the graph), the bond graph fragments do not model the components but the behavior of the components instead. (Note that the check valves are not modeled by the author [Martens 72], as the check valves do not contribute to

the phenomenon that the author is interested in, which is the resonant frequency of the system.) Thus, the capacitor in the lever corresponds to the compliance in the lever rod, while the capacitor in the bellows corresponds to the bulk modulus of the air in the bellows -- two distinctly different domains! However, the behavior of the capacitor in both circumstances is the same: it stores and gives up energy without loss. Mathematically, the energy of the capacitor can be defined as:

$$E(t) = \int_0^t e(t)f(t)dt + E_o \qquad (2.1)$$

where

$E(t)$ = energy stored in the capacitor at any time t

E_o = energy stored at time $t = 0$

$e(t)$ = effort

$f(t)$ = flow.

This is the phenomenon that the designer is modeling; and both the air in the bellows and the material in the lever rod exhibit behavior that can be described by the above mathematical equation.

In the lever, effort is the force and flow is the velocity. In the bellows, effort is the pressure difference and flow is the flow of air through the bellows.

Thus, bond elements and bond graph fragments represent different sub-systems and components in different domains, as long as their underlying behavior is similar. Table 2.1 clearly illustrates this property.

Another interesting feature of bond graphs is the networks that they form. These are possible due to the existence of junction structures.

2.4.3 Junction Structures

Junction structures, which are assemblages of

are the energy switchyards that enforce the constraints among parts of dynamic systems. No power is dissipated or generated in a junction structure, so the net power into a junction structure at its ports is always zero.

Junction structures provide the transformations of variables. The power invariance with the transformations enforced by junction structures enable junction structures to embody Kirchhoff's laws for electric circuits, equations of continuity and dynamic equilibrium in fluid systems, and Newton's laws and geometric compatibility constraints for a restricted class of mechanical systems.

If the mathematical model of the bond graph is maintained, the various first principles just described will be upheld. To partition a multiport system into distinct, interconnected fields, the bond graphs may be organized as shown in Figure 2.3. There are two types of bonds. External bonds join one member of the set (R,C,I,SE,SF) at one end, while internal bonds join only members of the set (0,1,TF,GY).

External bonds may be classified further according to the type of element they adjoin; namely, storage-field bonds adjoin C or I elements, dissipative-field bonds adjoin R elements, and source-field bonds adjoin SE and SF elements. Internal bonds are external to the junction structure of the system, and external bonds are the interface of the junction structure with the other fields.

As Figure 2.3 shows, the storage field is connected to the junction structure by storage-field bonds, the dissipation field is connected to the junction structure by dissipation-field bonds, and the source field is connected to the junction structure by source-field bonds.

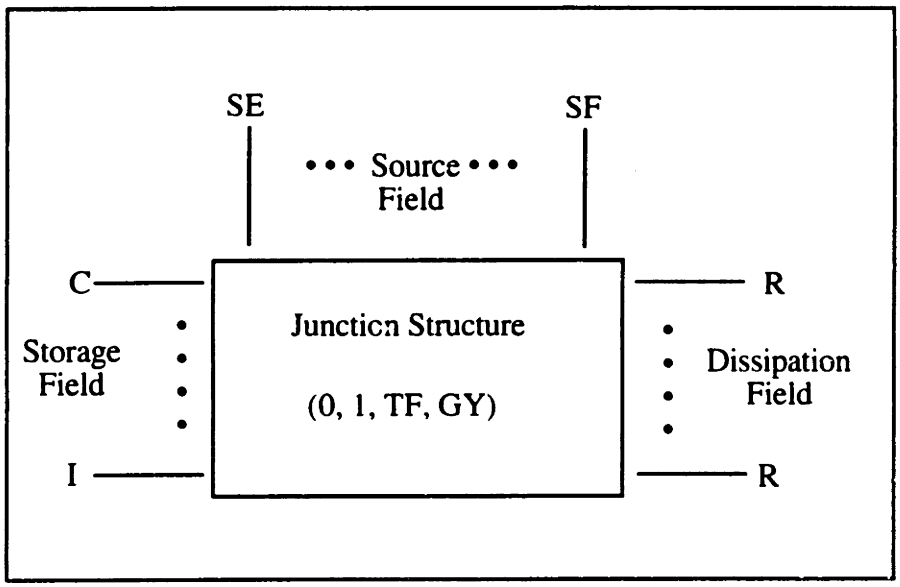


Figure 2.3 Basic fields of a multiport system

The first principles that are embodied in the junction structures are maintained when the graphs are manipulated within the junction structure. This is because the junction structure is isenergic, and therefore power is conserved. This property of the bond graph is exploited heavily when the transformation operators are applied to the representation.

2.4.4 Justification of the Choice of Bond Graphs

Bond graphs are used for the representation for a number of reasons. These include:

1. Uniform representation for different domains such as electrical, mechanical-translational, mechanical-rotational and hydraulic. It has been used in some thermal

systems by some researchers. It may not be suitable for two-phase, fluid and thermal systems.

2. Descriptive and graphic language that, once learned, will be able to perform the modeling intuitively.

3. Grammar-like properties [Finger 89], with the implication that the solution space represented by the bond graphs need not be enumerated, but can be generated by the set of elements in the bond graph grammar.

4. Mathematical properties that allow the evaluation of dynamic characteristics. Bond graphs are a practical device for the generation of differential equations, block diagrams, signal flow graphs and transfer functions for physical systems.

5. Ability to model nonlinear systems. This property allows many real-world systems to be modeled.

6. Ability to model multiple-input-multiple-output systems. This property allows the modeling of control systems and complex dynamic systems.

7. Common representation of the problem formulation and solution. These are typically thought of as entirely separate operations. Bond graphs enable the problem and the corresponding solution to be represented in the same operation. In other words, the problem is represented by the bond graph and the solution also, but with the configuration, parameters and values changed. One of the most impressive features of bond graphs is their ability to be transformed, from a rough existential statement about the sort of model of a physical system to be used, to a detailed graph that contains all the information required for a simulation of the system, by means of a series of operations on the original bond graph.

Other than the above-mentioned salient points of bond graphs as representation, bond graphs have been examined by the author to provide the following operator that may be vital for future work.

2.4.5 Analogy

In design, analogy is often used to generate alternatives. At first sight, bond graphs are able to provide a natural and apparently small set of analogous elements from the four domains listed in Table 2.2. However, networks formed from these elements and other network elements provide a host of sub-systems and systems that correspond to behavior in sub-systems and systems in the other domains. For instance, a network comprising a capacitance and resistor connected to a zero junction may correspond to a shock absorber with the damper and spring connected in series in the mechanical-translational domain. In the electrical domain, the network will correspond to a capacitor and a resistor connected in parallel. This, in turn, may be used as a charge-and-discharge circuit of a network to reduce voltage surges.

Thus, bond graphs can serve very effectively as a unified database, especially when devices and systems involve several energy domains simultaneously. In addition to providing a succinct, flexible database for linear and nonlinear, static and dynamic models, bond graphs can be processed causally to reveal important information about alternative input-output choices and device-level coupling factors when sub-models are assembled into systems.

This is indeed where the operator of the analogy can be utilized. The database of components and subsystems can be constructed using the corresponding bond graph as an index. When these components are required, the bond graph index is used to retrieve a set of corresponding items. These items from different domains are available for the designer to make the selection. Since the focus of this research is on the construction and

verification of the computational model for the automated generation of design variants, the analogy part of the research has been left for future work.

2.5 The Genetic Algorithm

2.5.1 Introduction

The genetic algorithm (GA) is an optimization strategy based on the theory of natural selection and natural genetics [Holland 1975]. As algorithms, they are different from traditional optimization methods in the following aspects:

1. GAs work with a coding set of the variables, and not with the variables themselves.
2. They search from a population of points, rather than by improving a single point.
3. They use objective function information without any gradient information.
4. Their transition scheme is probabilistic, whereas traditional methods use gradient information [Goldberg 1989].

Genetic algorithms are in general very robust. They are particularly suited to design problems involving large numbers of variables. Unlike other optimization techniques, the GA has the ability to avoid convergence upon local optima, and can successfully negotiate search spaces that are discontinuous.

2.5.2 Design Metric

Crucial to the success of the genetic algorithm is the choice of the design metric.

In natural selection, where survival of the fittest is the rule, design metric is the measure of the fitness of the organisms. The organisms, each of which represents a possibly optimal solution to the optimization problem, compete with one another to provide the best solution to the problem. Those organisms that are most highly fit are allowed to serve as parents, mating with each other to create child organisms. After undergoing random mutation, the child organisms replace the parents, and the evolutionary process iterates. Optimization occurs, therefore, when many generations of evolution improve the quality of the artificial organisms.

2.5.3 Coding and Decoding

An essential characteristic of a genetic algorithm is the coding of the parameters that describe the problem. The most common coding method is to transform the parameter to a binary string of specific length. This string represents the chromosome of the problem, and the length of the chromosome represents the number of zeros and ones in the binary string.

For a specific problem that depends on more than one parameter, a multivariate coding is constructed by simply concatenating as many single variable codings as the number of variables of the problem. Each parameter may have its own length corresponding to the minimum, maximum, and a step value specified for the particular application. By decoding the individuals of the initial population, the solution for each specific instance is determined and the value of the objective function that corresponds to this individual is evaluated.

2.5.4 Reproduction

A simple genetic algorithm proceeds by first randomly generating a population of a specific size. A random generator is used to generate the initial population. From this population, the next generation is evolved by performing three distinct operations: namely, reproduction, crossover and mutation. Based on the statistics of this population, the next generation is reproduced following the weighted-roulette-wheel method as the default method. Adopting a bias law, probabilities are assigned to the members, analogous to the statistics of the generation. This means that the weak designs will be assigned small probabilities and the strong designs will be assigned high probabilities of existence in the next generation. In this way, the next generation evolves where the fittest have survived, and increase their presence; while the weaker designs die out, or disappear from the generation.

2.5.5 Crossover and Mutation

In the process of the reproduction of the new generation, the operations of crossover and mutation are performed. With a specified probability of crossover, two members of the population, selected randomly, exchange part of their chromosomal information by exchanging the right part of their string at a randomly selected point. (This is shown in Figure 2.4). The probability of crossover determines whether crossover will be applied to the selected parents or not. This corresponds to a single-point crossover scheme. Alternatively, a multipoint crossover scheme can be used [Booker 87][DeJong 75].

In the process of mutation certain digits of the chromosome are toggled. (If the chromosome is not a binary string, the digits are changed.) If they are found to be zeros,

they are changed to ones, and if they are found to be ones, they are changed to zeros. After crossover and mutation, the population takes its final form in the current generation. Again, by decoding the strings and solving the sizing problem, new objective function values are determined. These objective function values express the fitness of the designs of this generation.

Simple statistics are deduced for this generation, and the process goes on by performing reproduction, crossover and mutation. After several generations, the best member of the population turns out to represent a very satisfactory solution of the problem. Theoretical evidence about the asymptotic convergence of the algorithm exists in the context of the methods of local search, simulated annealing, and Boltzman machines.

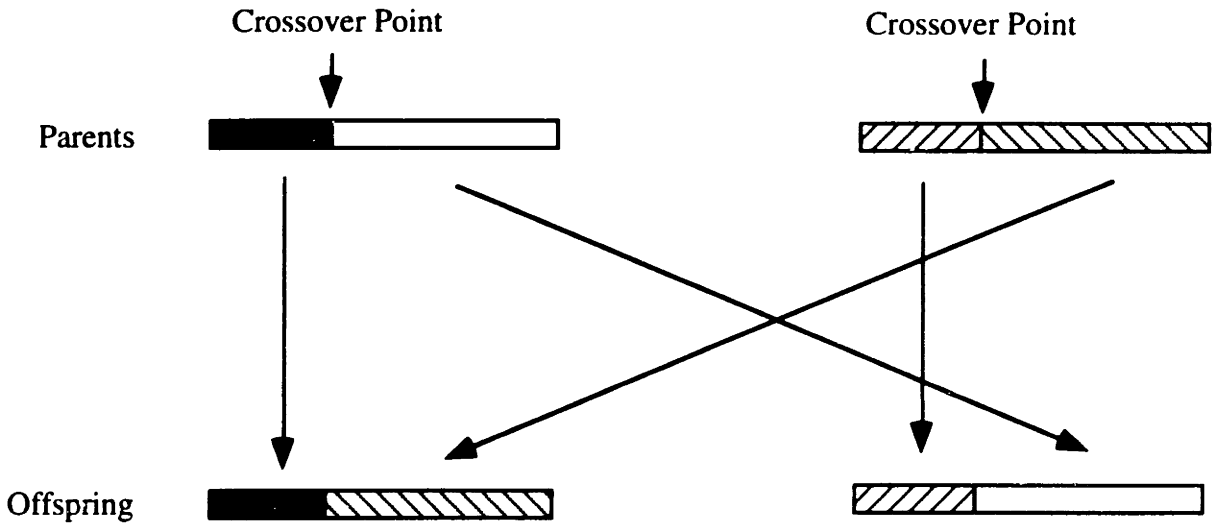


Figure 2.4 Single-point crossover

2.5.6 Basic Parameters of Genetic Algorithms

The basic parameters of a simple genetic algorithm are the population size of the generation, the probability of crossover, and the probability of mutation. By varying these

parameters, the convergence of the problem is altered. Thus, to maintain the robustness of the algorithm, it is important to assign appropriate values for these parameters. Usually, the population size and probability of mutation are related. With larger population size, the probability of mutation is smaller. For a wide range of problems, the following values are good estimates for an initial run: population size 30-50 or more, probability of crossover 0.8 and probability of mutation about 0.01. In general, mutation is important in the evolution of the method, because through mutation new members enter the population, bringing new blood to the generation. In this way, the initial population, which might have been very far from the satisfactory solution, can adapt itself toward the optimum solution. On the other hand, mutation tends to disorganize the convergence of the problem making the selection of this parameter, together with the population size, very crucial for the overall performance of the genetic algorithm.

Another way of improving the performance of the algorithm is to pass some of the best designs of the generation to the next generation. This, in effect, corresponds to a different bias law of reproduction for the next generation, which assigns greater probabilities of existence to the best designs of the population.

2.5.7 Speciation

Speciation is the formation of species, or clusters, in the population. To understand the utility of such a strategy, consider the action of a simple genetic algorithm on the simple one-dimensional sample function shown in Figure 2.5. With an initial population chosen uniformly at random, a relatively even spread of points across the function domain is obtained. As reproduction, crossover and mutation proceed, the population climbs the peaks, ultimately distributing most of the strings near the top of one hill. This ultimate

convergence on one peak or another, without differential advantage, is caused by genetic drift: stochastic errors in sampling, caused by small population size.

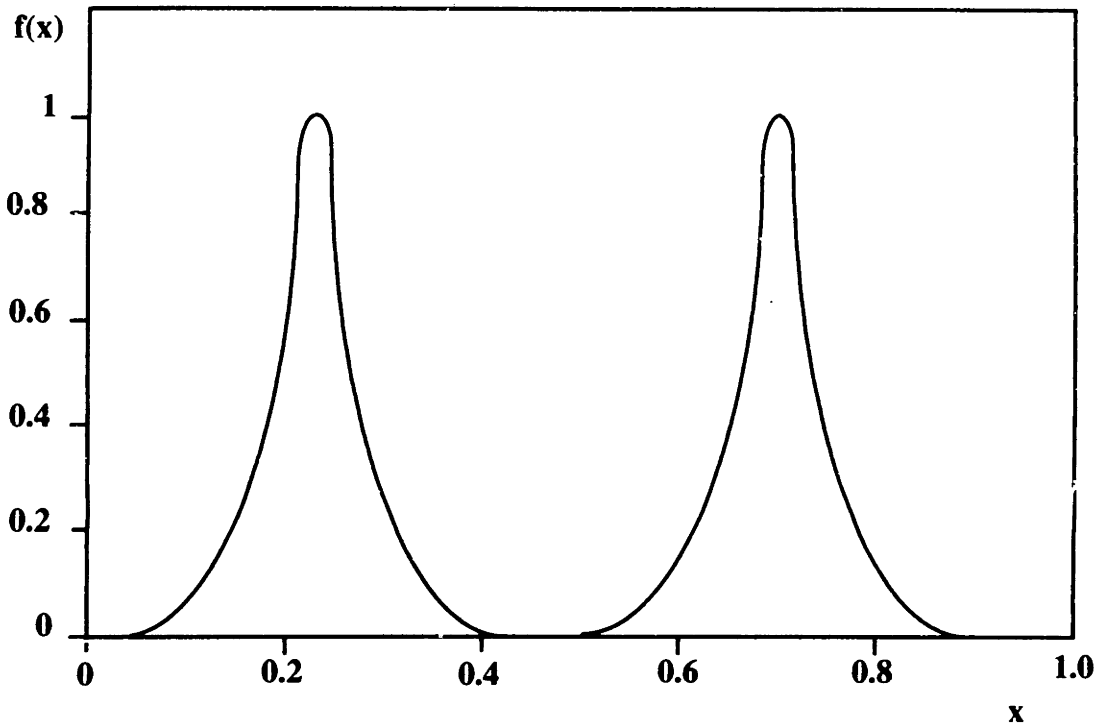


Figure 2.5 Sample function with many peaks

As the peaks are of the same height, the desired behavior is for stable sub-populations (species or clusters) to form around each peak. (This is shown in Figure 2.6.)

A practical scheme that limits the uncontrolled growth of particular species within a population is sharing [Goldberg 87]. In this scheme, a sharing function is defined as determining the neighborhood and degree of sharing for each string in the population. Consider the triangular sharing function shown in Figure 2.7. σ_{share} is a parameter that the designer can set to change the slope of the line.

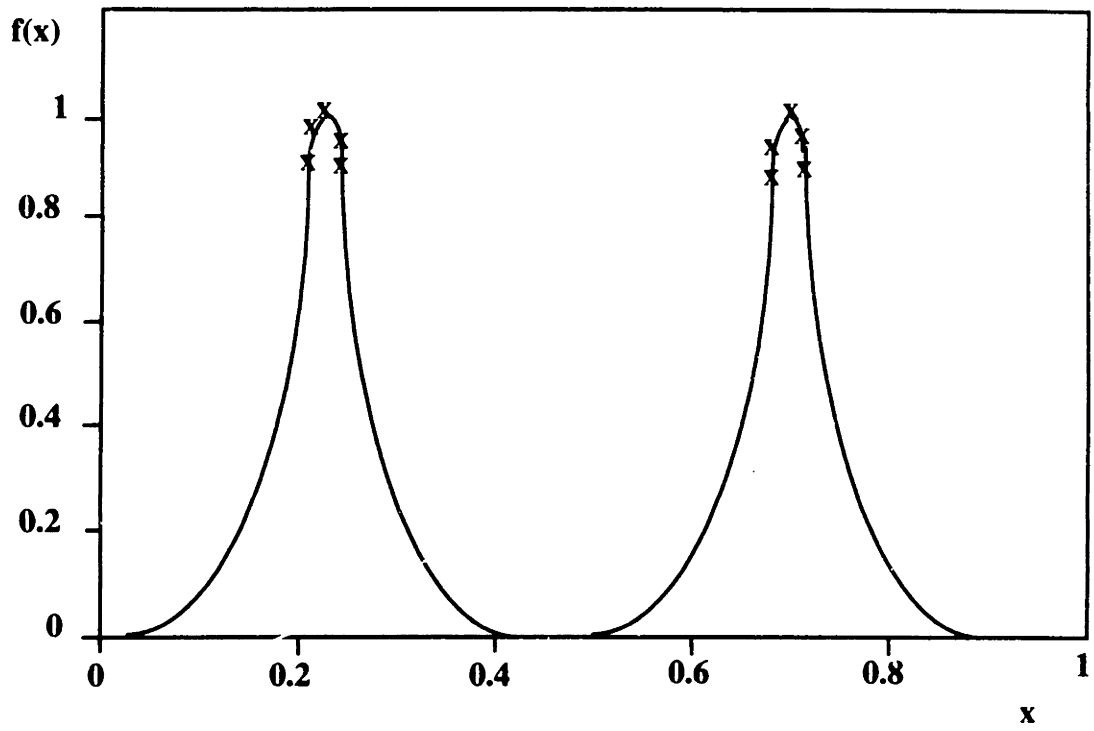


Figure 2.6 Simple genetic algorithm performance on equal peaks with sharing

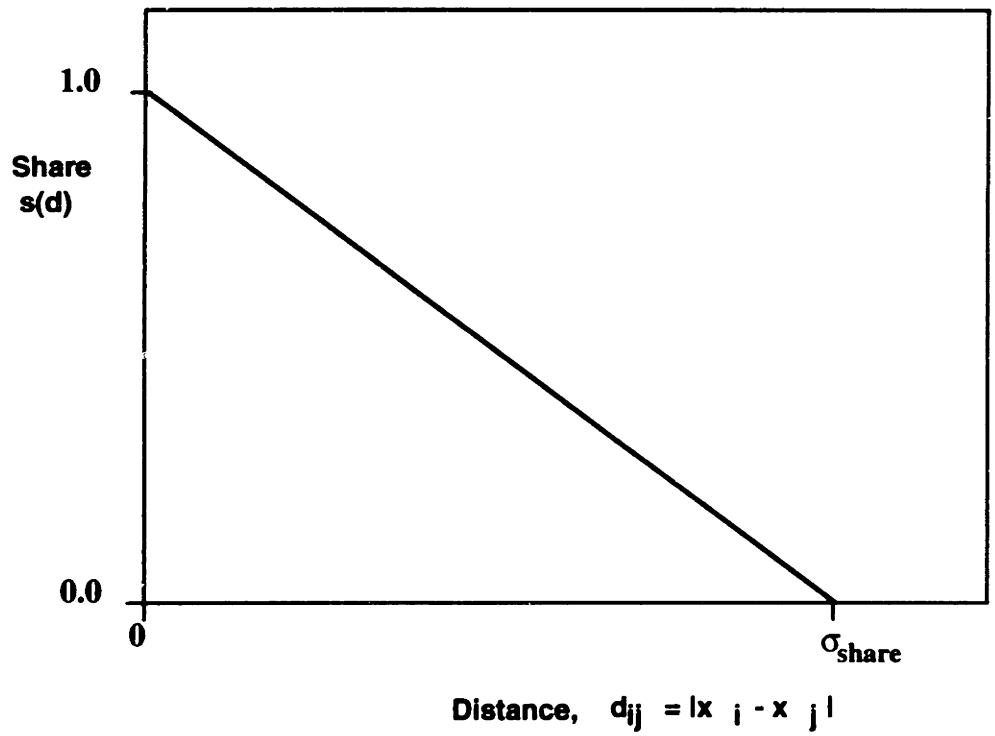


Figure 2.7 Triangular sharing function

For a given individual, the degree of sharing is determined by summing the sharing function values contributed by all other strings in the population. Strings close to an individual require a high degree of sharing (close to one), and strings far from the individual require a very small degree of sharing (close to zero). Since an individual is very close (as close as possible) to itself, its sharing function is one (as is any string identical to that individual). After accumulating the total number of shares in this manner, an individual's derated fitness is calculated by taking its potential fitness (the unshared value) and dividing by the accumulated number of shares:

$$f_s(x_i) = \frac{f(x_i)}{\sum_{j=1}^{n} s(d(x_i, x_j))} \qquad (2.1)$$

Thus, when many individuals are in the same neighborhood, they contribute to one another's share count, thereby derating one another's fitness values. As a result, this scheme limits the uncontrolled growth of particular species within a population. Instead, stable clusters (species) of individuals are formed around each peak.

2.5.8 Why do Genetic Algorithms Work?

The schema theorem [Holland 75] explains the power of the GA in terms of how schemata (which represent similarities between strings) are processed [Goldberg 89]. A schema is a string of total length l (the same overall length as the population's strings), taken from the alphabet {0, 1, *}, where "*" is a wild-card or "don't care" character. Each schema represents the set of all binary strings of length l, whose corresponding bit-positions contain bits identical to those '0' and '1' bits of the schema. For example, the schema, 11**0, represents the set of five-bit strings, {11000, 11010, 11100, 11110}.

Schemata are also called "similarity subsets" because they represent subsets of strings with similarities at a certain, fixed number of bit-positions. Two properties of schemata are their order and their defining length. Order is the number of fixed bit-positions (non-wild-cards) in a schema. Defining length is the distance between a schema's outermost, fixed bit-positions. For example, the above-mentioned schema is of order 3, written o (11**0) = 3, and has a defining length of 4, written d (11**0) = 4. Each string in the population is therefore an element of 2^l schemata.

Individuals in the population are given opportunities, often referred to as reproductive trials, to reproduce. The number of such opportunities an individual receives is in proportion to its fitness; hence, the better individuals contribute more of their genes to the next generation. It is assumed that an individual has high fitness because it contains good schemata. By passing more of these good schemata to the next generation, the likelihood of finding an even better solution is increased.

[Holland 92] showed that the optimum way to explore the search space is to allocate reproductive trials to individuals in proportion to their fitness, relative to the rest of the population. In this way, good schemata receive an exponentially-increasing number of trials in successive generations.

The schema theorem is included here for completeness.

2.5.9 Schema Theorem

The simple GA, before significant convergence, allocates an exponentially increasing number of trials to useful schemata. This is illustrated by the re-derivation of the schema theorem.

Let m(H,t) be the number of instances of schema H present in the population at generation t. The expected number of instances of H at the next generation, m(H, t+1) is calculated in terms of m(H,t).

In simple GAs, fitness-proportionate selection assigns each structure i in the population a probability selection $p_s(i)$, according to the ratio of i's fitness to overall population fitness:

$$p_s(i) = \frac{f(i)}{\sum_{j=1}^{n} f(j)} \qquad (2.2).$$

It follows that H can expect to be selected $m(H,t)\cdot(f(H)\bar{f})$ times, where $\bar{f}$ is average population fitness and f(H) is the average fitness of those strings in the population that are elements of H.

The probability that single-point crossover disrupts a schema is precisely the probability that the crossover point falls within the schema's defining positions (those outermost, fixed bit-positions used to calculate the defining length). The probability that H survives crossover is greater than or equal to the term, 1- $pc \times \frac{\delta(H)}{l-1}$. This survival probability is an inequality, because a disrupted schema might regain its composition if it crosses with a similar schema. The probability that H survives mutation is $(1 - pm)^{o(H)}$, which can be approximated as $1-o(H)\ p_m$ for small p_m and small $o(H)$. The product of the expected number of selections and the survival probabilities (with the smallest multiplicative term omitted) yields what is known as the schema theorem:

$$m(H,t+1) \geq m(H,t)\cdot\frac{f(H)}{\bar{f}}\cdot(1-p_c\frac{\delta(H)}{l-1}-o(H)p_m) \qquad (2.3).$$

The schema theorem states that low-order, short-defining-length, highly-fit schemata grow exponentially over time, while below-average schemata decay at a similar rate.

Chapter 3

Literature Review

3.0 Overview

This research builds upon previous work from several different areas, including bond graphs, representation of mechanical behavior, genetic algorithms, and conceptual design. In this chapter, these past works are reviewed in relation to the thesis developed. First, representation used in conceptual design is reviewed. The representation used in this thesis for mechanical systems is then discussed. Next, the design model on which the computational model is based is compared with the other design models used. Finally, previous design works implemented using genetic algorithms are reviewed. In all the works done so far, it should be noted, none has approached the design of dynamic systems using both bond graphs as representation, and genetic algorithms as the framework to embody the transformation operators.

3.1 Building Blocks of this Work

A summary of the work done thus far can be seen in Figure 3.1. Indeed, this work builds upon the foundation of design models that have been developed thus far. Though the design models are not formulated in a manner that can be computationally mechanized,

they serve as a good starting point for this research.

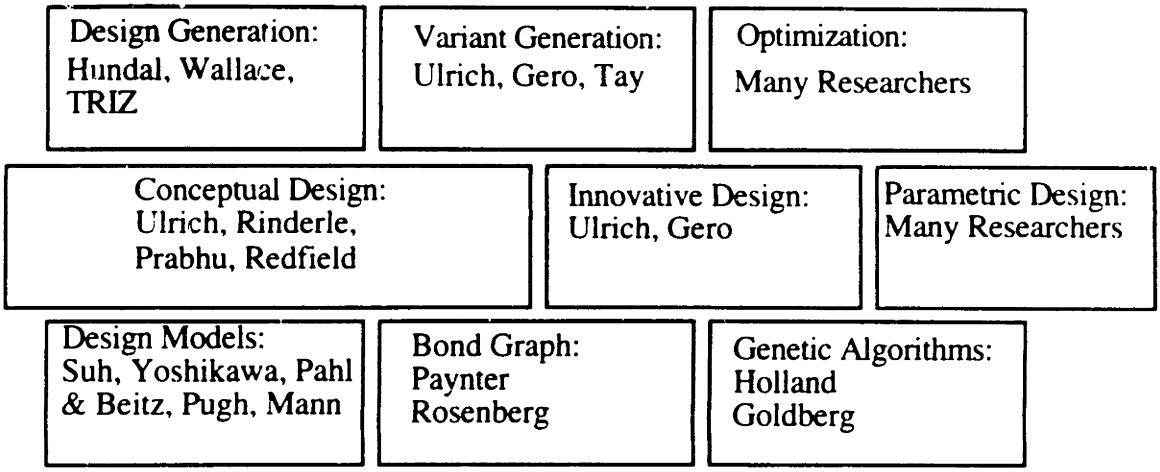


Figure 3.1 Building blocks of this research

Bond graphs are used as representation in this research, as they are inter-domain in nature. Some of the work [Ulrich 88][Prabhu 89][Rinderle 90] done on conceptual design has used bond graphs as representation but in general, it has not used them to determine dynamic characteristics. In this sense, bond graphs have not been used as mathematical models, which they are.

Genetic algorithms have been used to solve many engineering design problems, but these solutions have different representations and therefore are not universal. The representation used in this research is general enough to cover all dynamic systems that can be modeled as lumped-parameter problems. [Gero 90] has used a common representation for his static structural design problems. He has been successful in adding parameters and operators in his operational definition of innovative design. In this research, the parameters are added, subtracted or replaced with another, and the connections among the components are changed.

3.2 Representation

In this thesis, bond graphs are used to represent the dynamic model.

The bond graph models are mathematical models in the usual meaning of the term even though they may be represented by stylized graphs and computer display rather than the more conventional sets of differential equations. This mathematical aspect of the bond graph is fully exploited in the implementation of the computational platform.

3.3 Bond Graphs in Design

A significant difference between previous efforts and the present research is that many of the previous efforts focused only on the schematic aspects of bond graphs, rather than the mathematical model that those graphs represent.

Ulrich and Seering use bond graphs in their research of conceptual design [Ulrich 89]. They are concerned mainly with synthesis in the conceptual design process. As such, there is no analysis (in the mathematical sense) work done and the mathematical aspects of bond graph as a representation are not exploited. Further, the researchers limit themselv to single-input, single-output static systems that can easily be represented by acyclic bond graphs. Their model comprises two parts. The input to the first part consists of two partial bond graphs, one representing the input and the other representing the output. The restriction on the specification is that the input and output must be specified in terms of time integral, time derivative or direct proportion. The program then generates a feasible bond graph that connects the input and output bond graphs. The graphs are evaluated using a set of rules devised by the researcher to test whether the graph thus formed matches the static input/output relation. If the graph fails, a set of debug rules is employed interactively

with the designer to find a feasible bond graph solution. Once an acceptable graph is found, the designer develops a physical description by personally selecting known components from a small database of available components. At this point in the design process, concerns about functional sharing among the components of the design are addressed [Ulrich 88]. The final output of the entire process is a set of possible preliminary designs.

Prabhu and Taylor present a set of basic theorems for using a variant of bond graphs in design [Prabhu 88]. Though they do not make use of the mathematical properties of the standard bond graphs, these researchers exploit the graph nature of bond graphs. They describe a method for generating "power-flow" graphs, based on a functional specification of inputs and outputs in terms of effort, flow and power variables. Taylor and Prabhu point out that, while graph grammars may add formalism to design, they are based on an unguided search that is often inefficient and unproductive. Two ways are proposed to limit the generation of graphs: partitioning of the requirements to aid in the generation of minimally-connected graphs (in the hope of using a minimal number of components); and minimizing the power flow through the bonds of the graphs.

In [Prabhu 89], Prabhu and Taylor modify their previous work to consider problems with vector specifications. They approach this sequentially, first solving the scalar part of the problem, and then modifying the solution to account for the position and orientation. Two new graph elements are introduced, an orientation transformer and a position transformer, both of which are magnitude-preserving. A method is given for using these elements to "tune" the design to satisfy the vector requirements.

The consideration of vector quantities has brought about the need for "support"-type graph elements [Prabhu 90]. The previously-described graph grammar of Prabhu and

Taylor is further modified to include the "S-link" that performs a supporting function. Their work states that a change in the orientation of a flow variable causes an imbalance of momentum affecting the equilibrium of a system. All elements within a graph must therefore be supported directly or through other elements. A procedure is presented for adding supports to a graph, as well as for determining the type of support that may be required.

Hauck and Taylor [Hauck 90] present a mathematical formalism paralleling the work in bond graphs. This research is based on energy flow to inductive, resistive, and capacitive elements represented by surface and volume integrals (giving these elements abstract form). The formalism allows for the combination of element primitives to generate preliminary designs for energy flow-type problems. However, the researchers do not include dynamic characteristics of systems in their design variants.

The application of bond graphs to the specialized domain of gear transmissions has been explored by Finger and Rinderle [Finger 89], who show the importance of being able to represent both the behavior of an object and its physical characteristics. Bond graphs are used to represent the functional part of a problem specification (reduction ratio, speed, etc.). A specification bond graph is transformed by a set of graph-rewriting rules to generate graphs that represent feasible physical systems [Hoover 89]. Because the problem-specification graph represents the desired behavior of the design, behavioral-preserving transforms are used. These transforms are component-driven; that is, the graph is transformed so that known gear components can be used to solve the problem. An important feature of this work is the exploration of all the behaviors a component might have. For example, bevel gears can both reduce speed and change direction, while spur gears reduce speed and offset input and output.

Additional work with bond graphs, deriving equations of motion to aid in analysis of rigid body systems, is presented by Rinderle and Balasubramaniam [Rinderle 90]. A mass module has been built up from bond graph elements to model planar rigid bodies. Each mass module has three connection ports representing three different locations. Dynamic characteristics are not included in the requirements.

3.4 Design Model

Design models have been developed by many researchers in design. These models, in general, are for the entire design process, from problem-formulation to detail design, and do not give much insight into generating ideas.

[Mann 70] described the classical model whereby design is an iterative process, starting from problem-identification. The possible solutions are formulated and, in turn, evaluated. From the evaluation, any adjustments to the solutions are made, and again evaluated. This process reiterates until the desired design is found.

[Suh et al 77] developed the axiomatic method of design, which is fundamentally different from all other existing methodologies. Essentially, there are four characteristics of the method. They include:

1. The existence of domains.
2. Mapping between domains.
3. A hierarchical vector within each domain.
4. Two axioms in design: the independence axiom and the information axiom.

While this model formalized some of the practices observed in design work, the model and the mapping operation between domains are not operationalized. This fact

makes the process difficult to automate. The set of axioms is currently more of an aid in the process of design assessment or design evaluation, than in the process of design synthesis or generation.

According to the general design theory [Tomiyama 87][Yoshikawa 82,87], the design process is an evolution of metamodels. A metamodel is a model of a model, and is defined by finite attributes. If intermediate stages are introduced during the process of design, then each evolution or detailization at each intermediate stage makes up part of the design process. As the successive refinement proceeds, checks must be made along the way to see whether the evolved design satisfies the functional requirements.

[Pahl 84] describes design as a process of establishing "function structures" at different levels of abstraction. A required function is broken down into several sub-functions, and the final function structure is obtained by unambiguously recombining the sub-structures. A list of generally valid functions is given in terms of the changing and varying of components; and the connecting, channeling, and storing of energy, material and signal.

According to [Pugh 90], design may be construed as having a central core of activities, all of which are imperative for any design, irrespective of domain. Briefly, this core, the design core, consists of market (user need), product design specification, conceptual design, detail design, manufacture and sales. All design starts with a *need* that, when satisfied, will meet the *need*. From the statement of the need, a product design specification (PDS) must be formulated; once established, this acts as the mantle that envelops all the subsequent stages of the design core. The PDS acts as the control for the total design activity because it places boundaries on subsequent designs.

3.5 Genetic Algorithms (GAs) in Design

The research most relevant to this thesis is that done by Gero et al [Gero 94][Gero 93][Gero 92]. [Gero 93] uses genetic algorithms to implement his notion of exploration in design, which can be characterized as a process that creates new design state spaces or modifies existing design state spaces. New state spaces are rarely created *de novo* in design. More commonly, existing design state spaces are modified. The result of exploring a design state space is an altered state space. Gero concludes that exploration precedes search, and that it effectively converts one formulation of the design problem into another. To implement the exploration process, he applied the operators of evolution (combination, selection and mutation) on his representation, which is called a prototype. The prototype is a frame-like structure that contains (some) structural and material information. The exploration process involves adding a new variable for creative design. The application domain is structural design, and the frame-like prototype representation must be constructed to span the wide range of possible designs. This is in contrast to the grammar-like property of bond graphs, which have the advantage of not having to enumerate the entire space. Further, dynamic characteristics cannot be represented in a prototype representation in a meaningful way.

Genetic algorithms have been used as a computational tool for configuration design by many researchers [Mittal 89][Roston 95][Carlson 95]. Configuration design is the process of forming functional systems by assembling selected components from manufacturers' catalogs. Generally, the engineer performs this design process in two steps: the selection of a basic system configuration, and the subsequent selection of specific components for that configuration. A genetic algorithm approach to configuration design will allow the integration of these two steps. In all this research, the behavior of the system thus formed is not investigated, because the representation adopted by the

researchers does not allow them to model the behavior of the system.

[Wallace 94] used GAs to search through catalogs in the design for remanufacture. Though the main thrust of his thesis is to develop a computational platform to address the multi-objective nature of the design for remanufacture, he uses genetic algorithms to search through discrete ranges of parameter value represented in catalogs.

[Chapman 94] used GAs to evolve different cross-sectional areas of a cantilever beam subjected to a point load. The representation is a 2-dimensional matrix where a value of one represents material and a value of zero is void of material. In this way, the researcher was able to investigate the optimum structure for a point load.

3.6 The Research in Perspective

The research, in relation to the work just described, is situated comfortably in the computational-conceptual design area. Though bond graphs are used as representation, they are different from most of the research done, as the mathematical model of bond graphs is utilized. This is in sharp contrast to earlier works [Ulrich 88][Rinderle 90] wherein bond graph representation is used as a schematic, albeit an inter-domain schematic.

This research is also different in another fundamental way: it uses operators rather than rules [Prabhu 89]. Operators are more generic in their applications and rule-based approaches are more parochial in scope, as only the area covered by the rules is applicable.

Further, the operator-based approach allows the designer to explore an area with which he is not familiar. The designer may not know the area well enough to develop rules for generating design variants.

In the area of design models, the computational model developed differs in a few areas: representation, operators, and design metric. That is, a computational model needs to have a more formal description of each of the components, so that it can be computationally mechanized. The computational model allows automation of the process that it models.

In the area of genetic algorithms, this research contributes to the application area of dynamic systems. The computational platform thus created allows the search of a wide selection of dynamic systems before physical embodiment. Though this work draws inspiration from Gero's work, it differs in many ways. First, the application and representation are different. Further, while Gero seeks to replace design-as-search model with "design as explore-and-then-search" model, the computational model developed in this thesis enhances the design-as-search model, although the space in which the search is performed is no longer the same. The design-as-search model is described in the next chapter.

Chapter 4

Development of Model for Innovative Design

4.0 Overview

The development of the model for innovative design as the the search in the problem-solution space is described in this chapter. Section 4.1 describes the design-as-search model. Next two strategies for searching – behavioral transformation and structural transformation – are developed. The concept of solution space is illustrated by an example. Finally, the concept of solution space is extended to the proposed problem-solution space.

4.1 Design as Search

Complex processes such as invention, decision-making and problem-solving have always been challenging for design theorists and design practitioners to analyze. One particularly useful tool, elaborated by Newell and Simon [Newell 70], was the notion of search in a solution space. An extension is now made to include the problem-formulation in the solution space; the space thus formed is called the problem-solution space. The problem-solution couple consists essentially of a set of requirements and a design proposed in response to it. The main implications of such a notion (design-as-search in the problem-

solution space) are: there must be appropriate representations of the problem-solution space; and there must be operators to move the state of the problem-solution couple from initial to final.

The problem-solution couple is illustrated in Figure 4.1. {Pr, So} is a complex consisting of a set of functional requirements, Pr, and a proposed solution, So. If there is more than one solution, there will be multiple couples: {Pr, S_1}, {Pr, S_2}, ... The design is subjected to a design metric: So*-So, which may or may not lead to an evaluation of the functional requirements: Pr*-Pr. The discrepancy ΔSo (and/or ΔPr) is identified, and may result in modifying the design, the requirements, or both – thereby producing {Pr+ΔPr, So+ΔSo}.

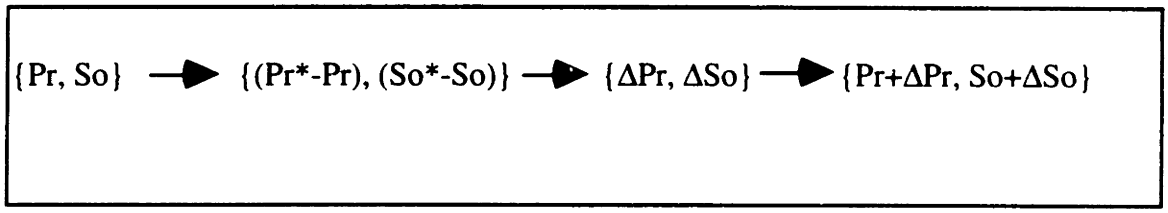


Figure 4.1 Transforming the problem-solution couple

The problem-solution space can be described in state space representation as in Figure 4.2.

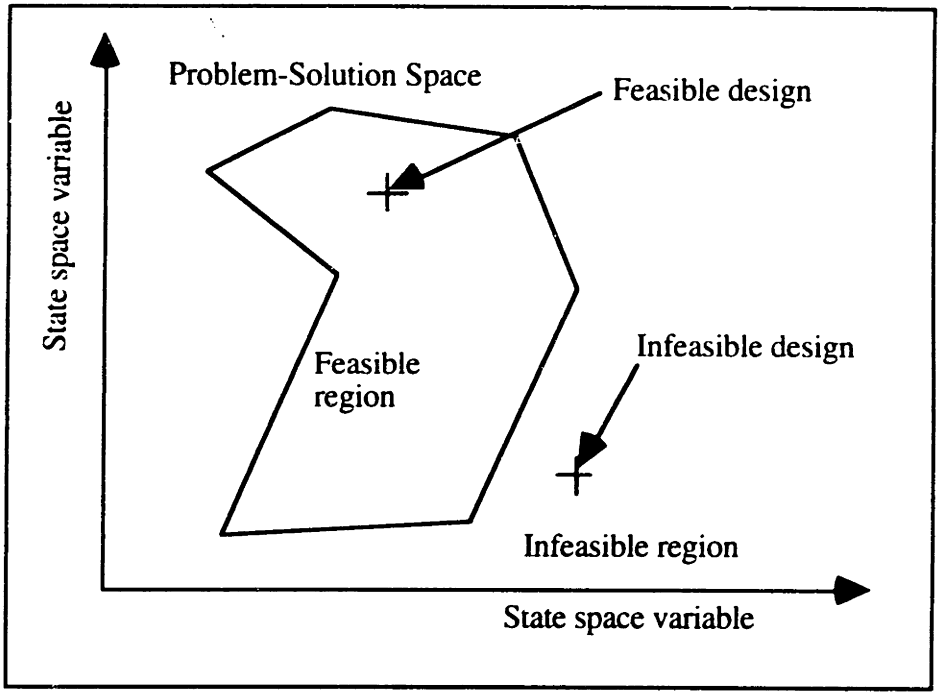


Figure 4.2 Design space representation of the problem-solution space

In a typical design search, owing to a lack of complete knowledge, the problem-solution couple may move in and out of the feasibility region. The problem-solution space is commonly characterized by a large infeasible region and small, disjointed feasible regions. This is shown in Figure 4.3.

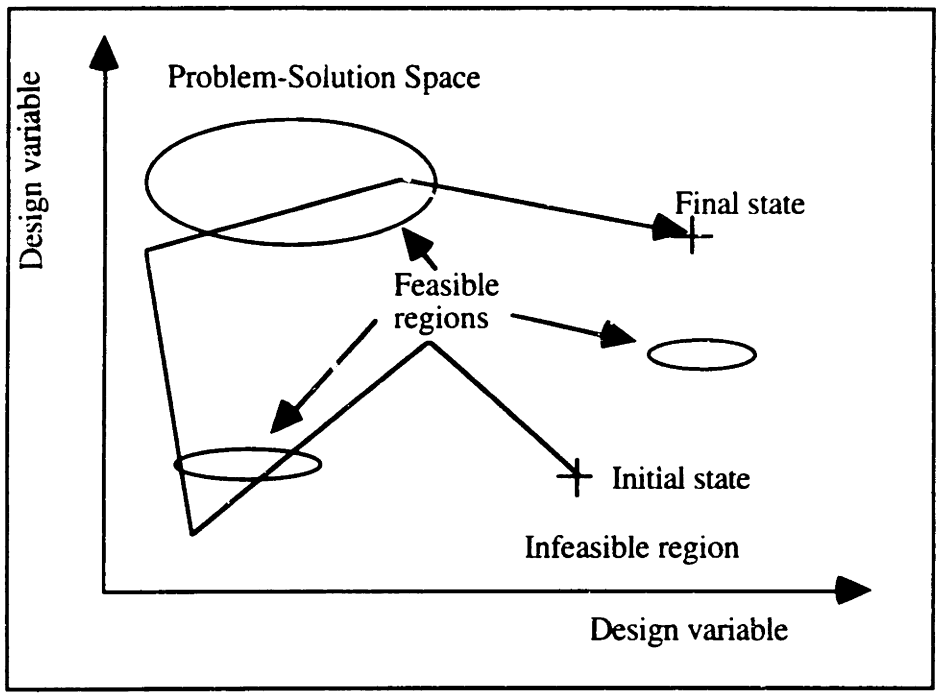


Figure 4.3 Design search

Based on the state space representation, the categories of design can be divided into parametric design and innovative design. In parametric design, the problem-solution space does not change, though there is a variation in the instantiation of the parameters represented by the state variables. In innovative design, the problem-space changes, since the state space variables used to describe the space are altered. (This is illustrated in Figure 4.4.)

There are two categories of operators: those that change the problem-solution space, and those that move the problem-solution couple within the space. The operators included in the first category are analogy, first principles, mutation, and combination. The second category includes operators such as selection and combination. As can be observed, some operators such as combination can change the state of the problem-solution couple and alter the feasibility region of the space.

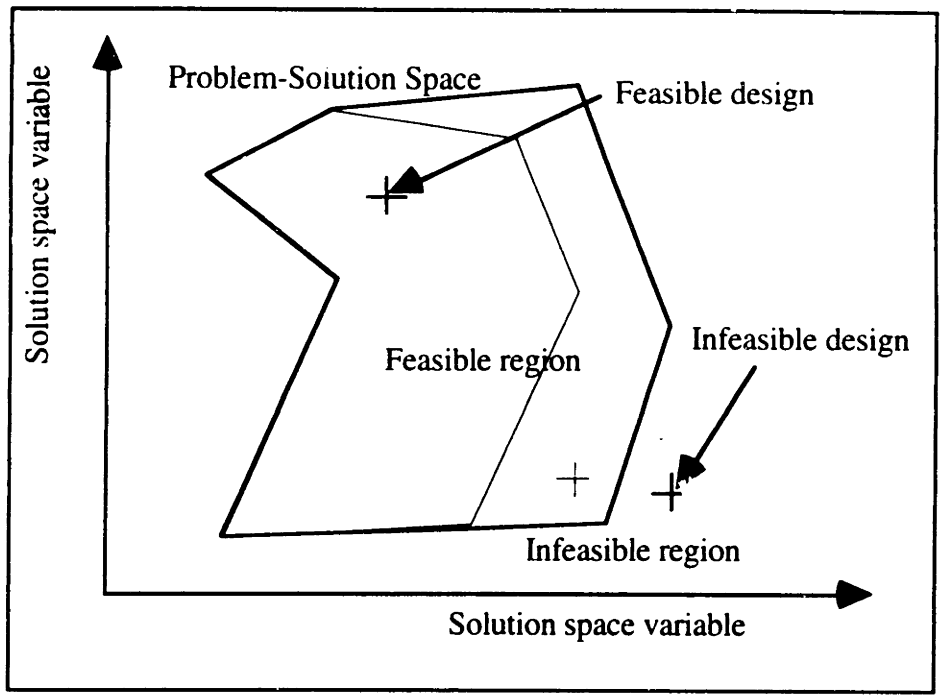


Figure 4.4 Innovative design

4.2 Strategies for Searching

Within the design-as-search paradigm, the overall process of design comprises four phases: formulation, synthesis, evaluation, and modification. Formulation involves the identification of the functional requirements of a problem. Synthesis includes the identification of one or more design solutions that satisfy the requirements. Evaluation involves the interpretation of partially-specified or completely-specified design descriptions for conformity with desired performance. Modifications can be made to the set of functional requirements, the proposed design, or both. The process is iterative.

The focus in this section is synthesis. The strategy adopted is transformation. (Other strategies include case-based reasoning and decomposition.) Transformation is a holistic approach to design. The transformation approach follows a theoretical approach to

design in which the initial set of design requirements is transformed into a design solution. The transformation strategy begins to approach the issue of how transformation occurs. Because grammars provide a formalism for the transformation model, they are used here to define the strategy.

A grammar is more commonly associated with language than with design. Understanding grammars and their application to design requires considering design knowledge as a language, and legal design solutions as legal statements in a language. In this sense, we can consider a grammar to be a formalism of design knowledge that can be used to generate a set of legal design solutions or, alternatively, to determine whether a design solution is legal. The term "legal" is used here in the broadest sense, meaning that a design solution or statement conforms with the formal definition in terms of the language.

The properties of grammatical formalism were explored by [Chomsky 57]. A formal definition of a grammar is

$G = \{N, T, P, S\}$,

where N is a set of non-terminal symbols, T is a set of terminal symbols, P is a set of production rules, and S is a special symbol called the start symbol.

In this research, the set of production rules is replaced by a set of design operators. The issue related to rule-based design methods is how rule-execution is controlled. This issue has further implications involving the construction of the set of rules. The choice of one rule over another may have to be made arbitrary, as the designer of the system may not know how the design will evolve. This is especially true for innovative design in which the problem-solution space is extended.

On the other hand, operator-based systems have the advantage of robustness. They are robust, as operator-based systems do not make any assumption of the problem-solution space. The space can be disjointed, with no necessity for continuity and monotonicity.

Bond graphs are used to define a grammar that captures the behavior of mechanical systems. The non-terminal symbols are C, I, R, SE, SF, TF, GY, 1, 0, and the bonds. The terminal symbols are chosen by the designer to be the input and output of the system, including the state variables. Alternatively, the terminal symbols may be the effort or flow of the elements: C, I, R, SE and SF.

The attractive aspects of a grammatical approach to design are the ability to represent a design space without enumerating possible design solutions, and the ability to apply the transformation strategy to both the behavior and the structure of the graph.

4.2.1 Behavioral Transformation

The approach adopted to generate design variants computationally is to transform the behavior of the system, while maintaining its function. The difference between function and behavior is the key to understanding the strategy. (This is illustrated by the two types of clocks in Figure 4.5.) In both cases, the function (defined by qualitative physicists as "what the device is for") is to tell time. Behavior, defined as "what the device does," is very different. One design rotates its hands, while the other increments its digits. A sundial and an hourglass can be added to the list of design variants satisfying the same function of telling the time. Yet their behaviors differ considerably. Therefore, changing the behavior while keeping the function constant is akin to generating design variants.

Furthermore, a device may have more than one behavior which can be exploited to serve more than one function. For example, a beverage can has the behaviors of enclosing a volume and supporting a mass, among others. These behaviors are exploited by its function of containing the beverage, and at the same time, of being stackable, one on top of another.

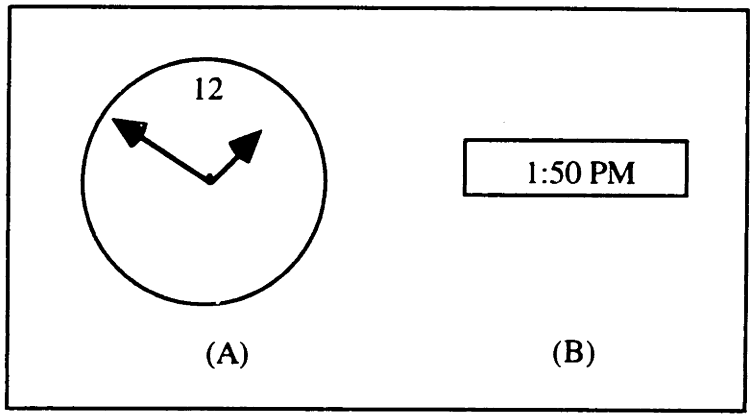


Figure 4.5 Two (2) behaviors for the same function

To change the behavior of the system, the behavior and the function of the system must be represented. This is done using bond graph grammar. Bond graphs are selected as a representation tool since bond graphs represent behavior, effect and phenomena, rather than objects. Further, the attractiveness of a grammatical approach to design is the ability to represent a design space without enumerating the possible design solutions.

The transformation of behavior is achieved by changing the structure of the graph.

The formulation is best illustrated by an example. The objective is to generate options for the function of transforming fluid pressure into force (see Figure 4.6). The original idea is converted into a bond graph. With this as the starting point, two things can happen:

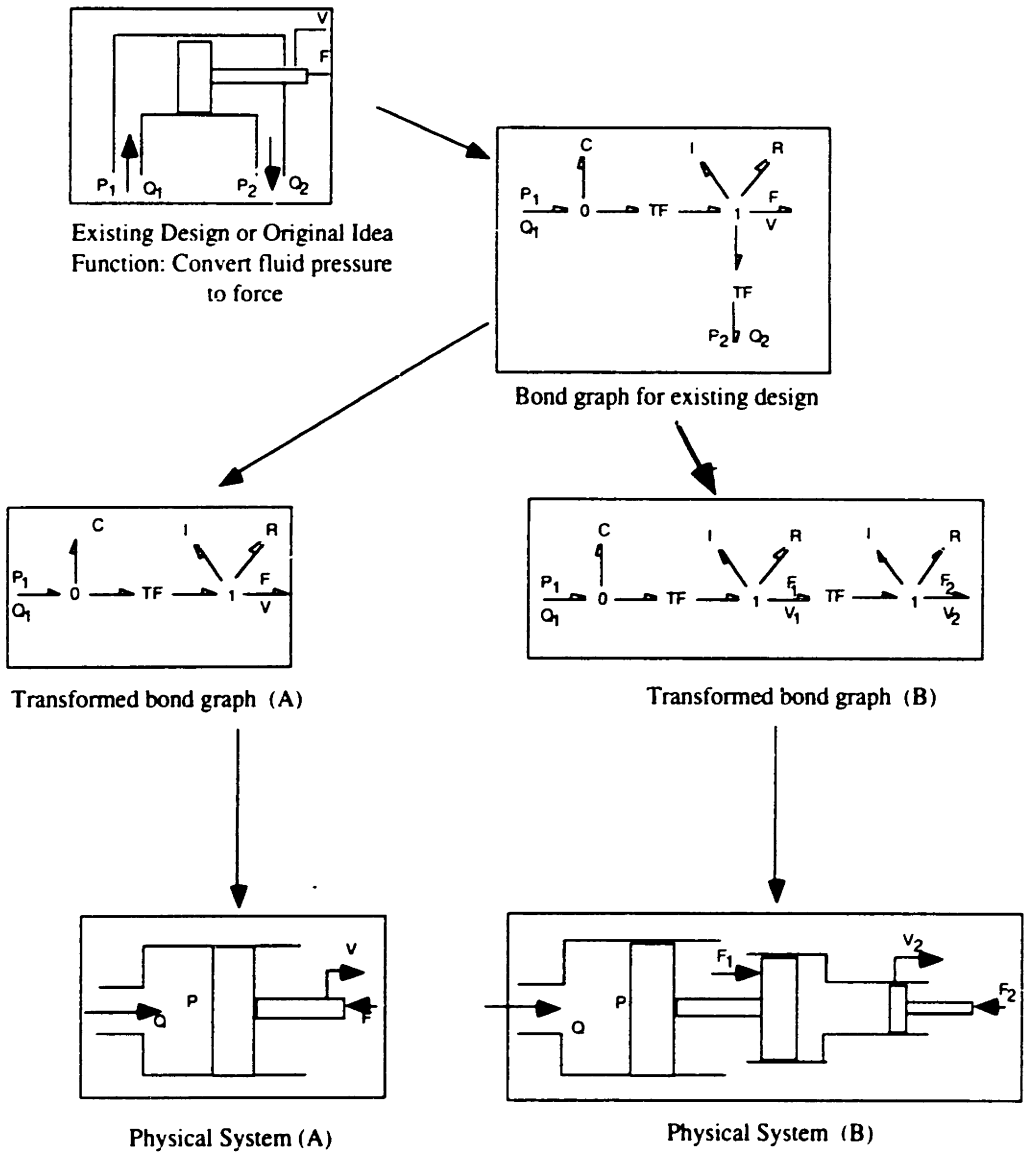


Figure 4.6 Behavioral transformation

1. The bond graph is modified topologically. The modified bond graph will then be mapped back into the original domain. This topological change corresponds to a behavioral change to the original system, in the sense that what the system does is different. For instance, there is no fluid return in Physical System A. Further, in Physical System B, the distance to be moved by the first piston is reduced. Note that all three systems (the original and the two design variants) serve the same function of converting a fluid pressure to a translational force.

2. Schemes represented by the branches of the bond graph and the terminal nodes of the bond graph are replaced, giving the structural transformation of the design. (This will be elaborated in the next section.)

4.2.2 Structural Transformation

The transformation of the structure of the design is achieved by fitting schemes into the branches and elements of the bond graph, as shown in Figure 4.7.

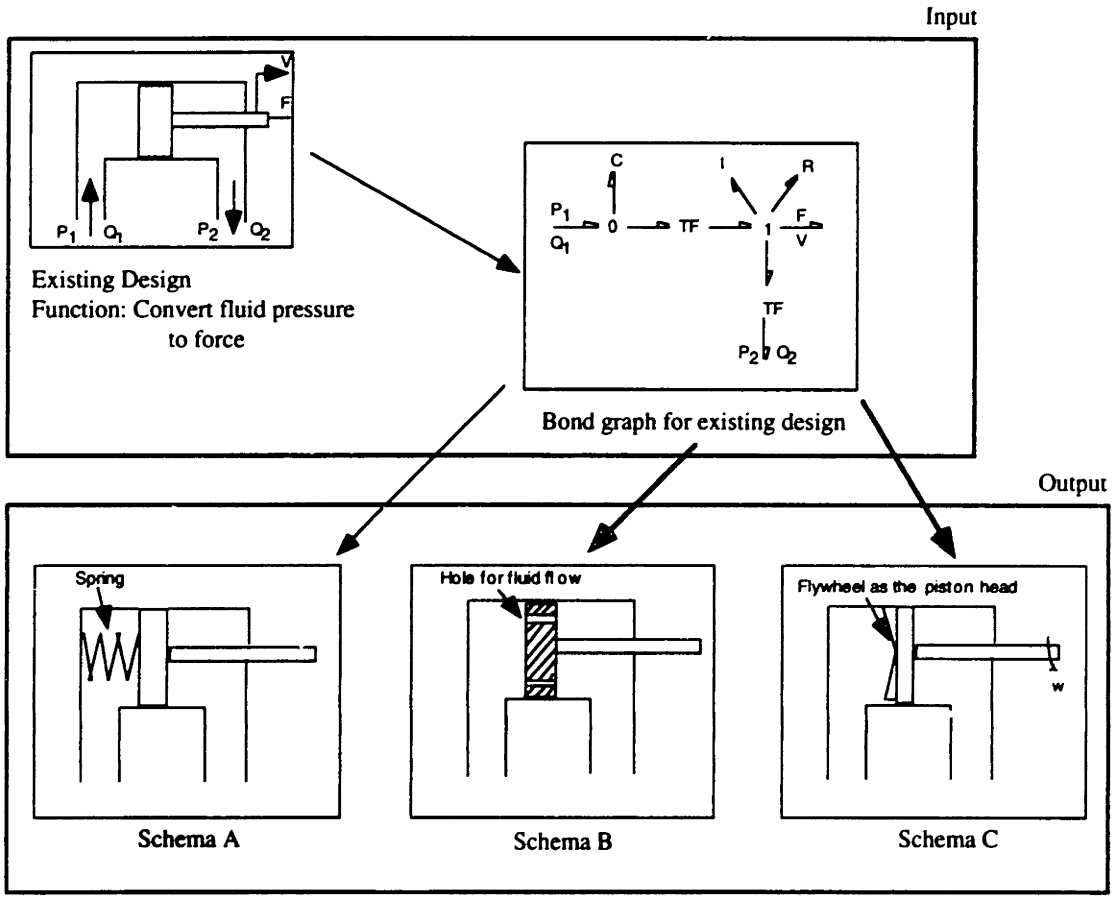


Figure 4.7 Structural transformation

With the input as shown, the various schemes are fitted into the branches of the graph. The capacitance represented in the bond graph is replaced in the first instance by a

helical spring. The resistance due to leakage in the original instance has now been made into a feature by cutting holes into the piston head, giving a more predictable force-loss relationship. In Schema C, the inductance is replaced by a flywheel! Though an apparently ludicrous idea, this may lead to insight in the final design.

4.3 Searching in the Solution Space

The solution space comprises all the solution that are described. In design, this is only possible in parametric design, as the assumption is that all the parameters are known, and the task is to determine the values of the parameters that will yield feasible solutions.

The understanding of the solution space is the key to developing a model of innovative design. An example is used here to illustrate the concept.

4.3.1 The Beam Problem

Consider the problem shown in Figure 4.8.

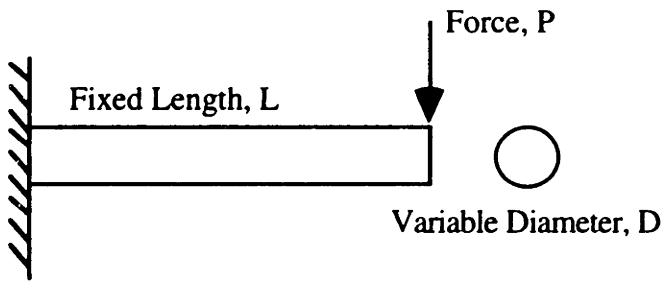


Figure 4.8 The beam problem

The objective of this problem is to minimize the deflection of the beam at the point where the force acts. Initially, we consider a circular cross-section. That is, the beam is a rod of diameter D. The constraints are minimum weight, bending stress, and shear stress. In a typical problem formulation, the problem is stated as shown in Figure 4.9.

min f(x) = deflection of rod = $64PL^3/3E\pi D^4$

subject to:

weight constraint: $\rho\pi D^2L/4$	<	maximum weight
bending constraint: $PL/128\pi D^3$	<	allowable bending stress
shear constraint: $16P/3\pi D^2$	<	allowable shear stress
non-negativity constraint: D	>=	0

Figure 4.9 Problem formulation

Putting some values into the parameters:

Length of the beam, L = 10m

Point load force, P = 14000N

Density of material, ρ = 7850kg/m3

Allowable bending stress, σ_b = 165MPa

Allowable shear stress, τ_b = 50MPa

Allowable weight, w = 80N.

For the rod:

Weight constraint:

$\rho\pi D^2Lg/4$ < weight

Substituting the values, D < 0.0114m.

Bending constraint:

$\sigma = PL/128\pi D^3 <$ allowable bending stress = 165MPa.

Wth the values, D > 0.0128m.

Shear constraint:

$\tau = 16P/3\pi D^2 <$ allowable shear stress = 50MPa.

With the values, D > 0.0128m.

Thus, as far as parametric design goes, there is no feasible solution. This can be illustrated on the number line shown in Figure 4.10.

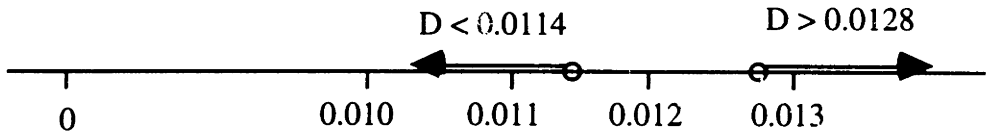


Figure 4.10 Null solution space

4.4 Searching in the Solution-Problem Space

The solution-problem space is obtained by extending the solution space. This can be achieved by changing, removing or adding parameters to the original problem formulation, thereby changing the original problem formulation. Before the general case of changing parameters is discussed, the particular case of adding a new parameter will be considered in order to illustrate how the addition of a new parameter can change the solution space. (In this case, we have introduced a solution space in which no feasible solution existed earlier.)

4.4.1 Addition of a new parameter

To expand the solution space, an additional parameter, the inner diameter, is introduced, in effect changing the problem formulation. The inner diameter has increased the solution space of nullity to that of a feasible region. (This can be seen in Figure 4.11.) This has also allowed the possibility of optimization.

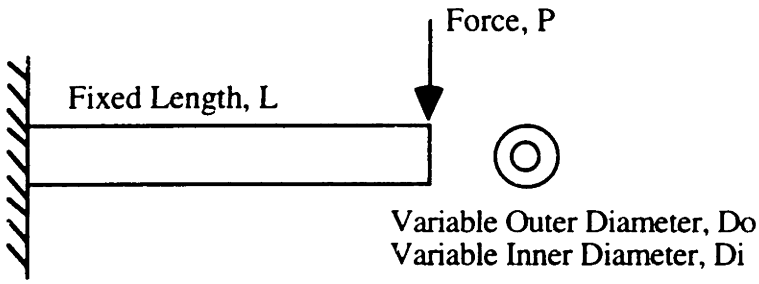


Figure 4.11 Addition of a new parameter

The problem formulation has changed. This can be seen in Figure 4.12.

min f(x) = deflection of hollow circular beam

subject to:

weight constraint: weight < maximum weight

bending constraint: bending stress < allowable stress

shear constraint: shear stress < allowable stress

$0 < Di < Do <$ Outer Limit

$Di >$ Inner Limit

Figure 4.12 Problem formulation for the addition of a new parameter

The problem is different from the original one, although the function is the same: that is, to minimize the deflection of the beam.

For this problem, with the replacement of a tube for the rod, inserting the values of the parameters:

$$\min \quad PL^3/3EI = 2.377x10^{-4}/(D_o^2-D_i^2)$$

subject to:

Weight constraint:

$$w = 2.466x10^5(D_o^2-D_i^2)g < 80$$

Bending stress constraint:

$$1.426x10^6D_o/(D_o^4-D_i^4) < 165x10^6$$

Shear stress constraint:

$$1.188x104(D_o^2+D_oD_i+D_i^2)/(D_o^4-D_i^4) < 50x10^6$$

Geometric constraint:

$$Do > Di$$

The above constraints and objective function are fed into an optimization package called OptdesX™, a software for optimal engineering design. The solution space is shown in Figure 4.13.

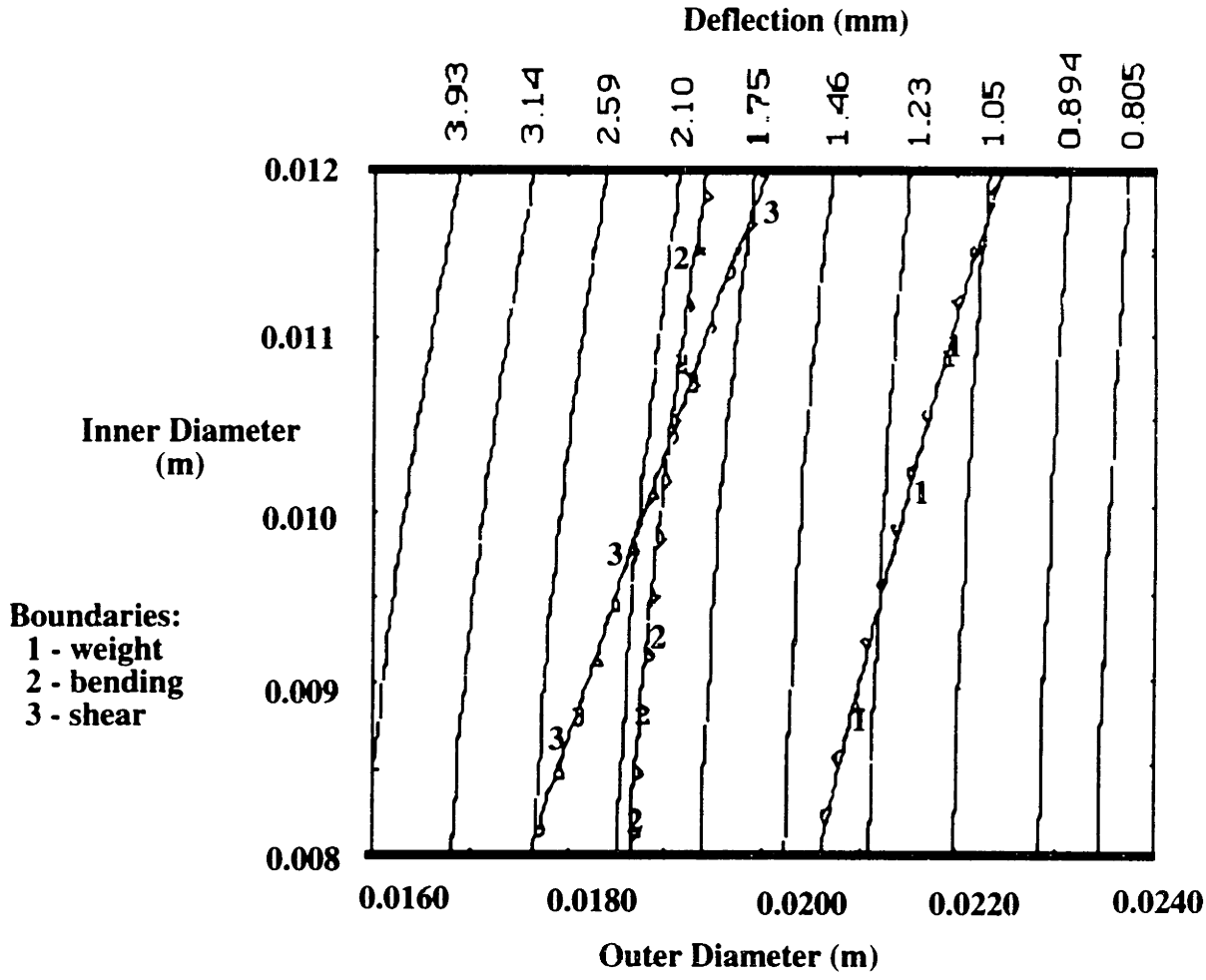


Figure 4.13 Solution space for the beam problem

4.4.2 Changing the parameter

In general, the solution space was changed by changing the parameters; that is, the addition of a parameter, the inner diameter in the above case, is just one instance of a number of changes that can take place by looking at the parameters in general. Thus, by changing parameters, such as length or cross-sectional area, the solution space is again altered. This is shown clearly in Figure 4.14.

Note that changing the problem formulation has led to the generation of many solutions that were not possible initially.

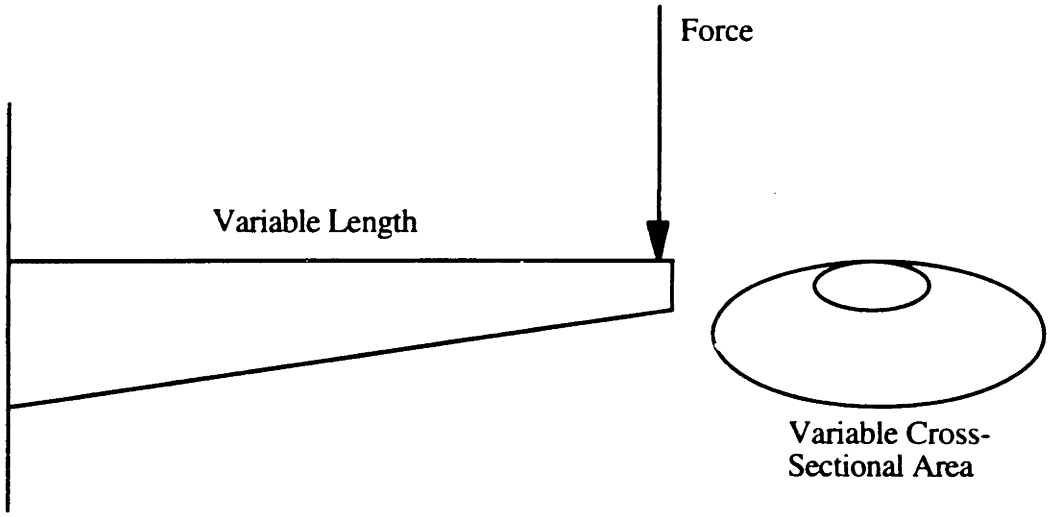


Figure 4.14 Changing the problem formulation

Thus, changing the problem formulation has enlarged the solution space. This has led to the realization that innovative design can be modeled as "search in the problem-solution space."

4.5 Innovative Design as Search in the Problem-Solution Space

Innovative design is defined as a design process, beyond the domain of parametric design, in which designs are not created by changing the values of parameters.

Modeling innovative design as search has many connotations: Design knowledge can be expressed as goals and operators. The problem-solution space must be represented. In turn, this means that design formulation involves the identification of the goal(s) of the

design. Design synthesis involves the search for design solutions through the selection and application of operators in the problem-solution space. Design evaluation involves assessment, if the goals are satisfied. (This is illustrated in Figure 4.16.)

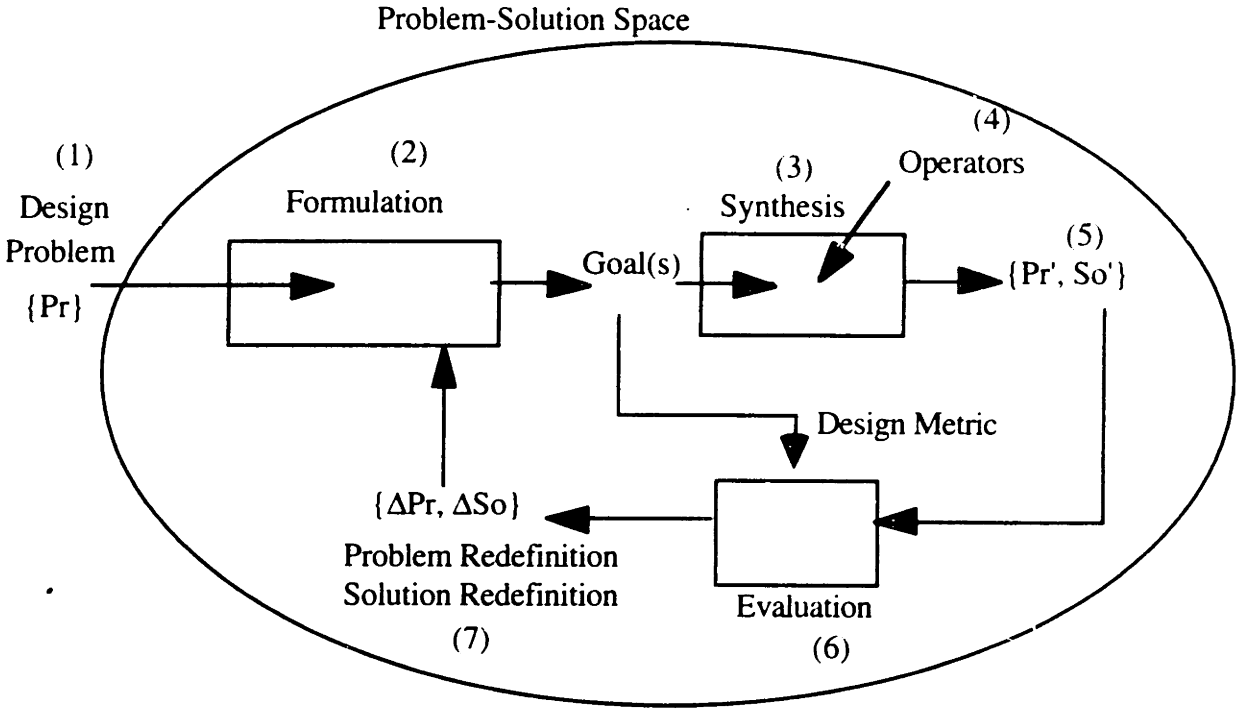


Figure 4.15 Model for innovative design

In the model for innovative design, the design problem (1) is specified in solution-neutral language. In the beam example, the problem may be specified as finding a structure with fixed span 1.2m that will not deflect more than 0.03m when subjected to a point force of 50N.

In the formulation (2) phase, the problem is described in a manner that facilitates the changing of the parameters. At this stage, the goals are encoded in a form meaningful and useful for the evaluation (6).

In the synthesis (3), operators are used to search for the solution. The operators also work on the problem formulation, extending the solution space to become the problem-solution space. The transformed couple {Pr', So'} is then evaluated. If the need arises, owing to a discrepancy between the design metric and the proposed solution, the problem may be redefined and reformulated and/or the solution modified (7).

The key distinguishing feature in this model is that the formulation of the problem is included in the model, and is not taken to be a given, as in many design models.

Chapter 5

Representation and Formulation

5.0 Representation in General

One of the remarkable attributes of human intelligence is the ability to convert a problem into a familiar form, or representation, that can be manipulated using previously-known techniques.

This section will examine the concept of representation: in particular, those representations that can be formally defined, and are thus suitable for computer mechanization.

The primary distinguishing characteristics of a representation are:

1. The information which is made explicit.
2. The manner in which the information is physically encoded.

The purpose of a representation is to simplify the problem of answering a restricted set of questions about a given situation. The selection of the representation must therefore be goal-directed.

The operations and data structures provided by the representation should result in simple computational procedures for answering questions relevant to the given situation.

5.1 Bond Graph Representation

One of the most important contributions of the bond graph representation is that input-output decisions for device models may be deferred until they are assembled into a system configuration. The graphical-energy nature of representation allows us to store the models in an acausal form, just as a circuit diagram does. That is, for a circuit model we are not required to decide *a priori* at a terminal pair which of the variables (for example, voltage, current) will be the input and which the output; similarly, at a port we are not required to decide *a priori* which of the power variables (for example, torque, angular velocity) will be the input and which the output.

Refer to the air pump shown in Figure 5.1.

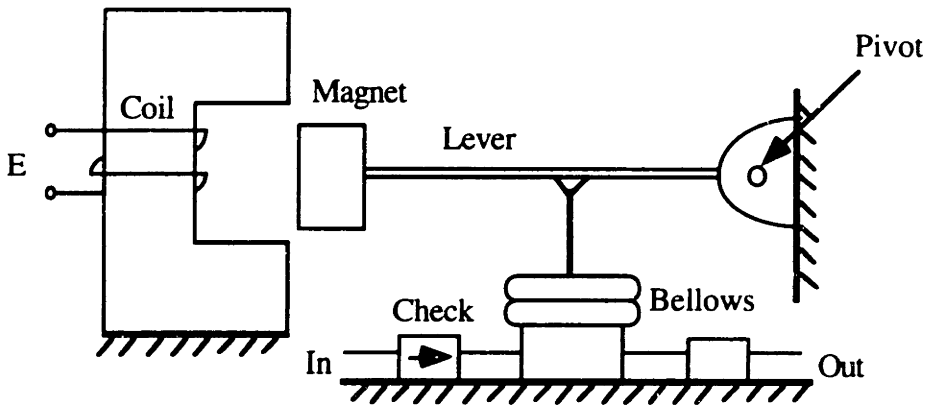


Figure 5.1 Schematic of air pump

The corresponding bond graph representation is as follows:

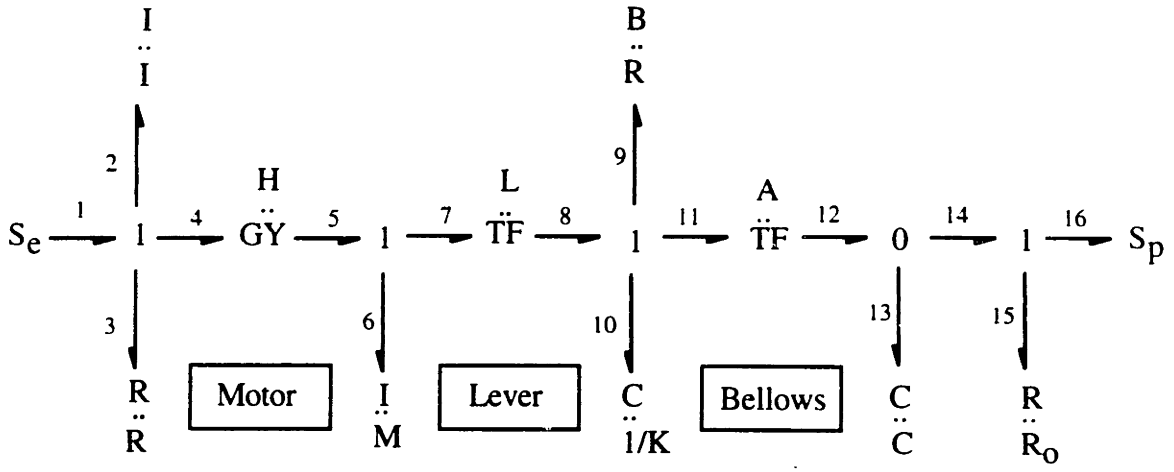


Figure 5.2 Bond graph for the air pump

The representation of the air pump, in the form of a bond graph, allows one to manipulate the air pump system by manipulating the graph. This is true, in general, of most forms of representation; and indeed, this is one of the most important functions of representation. For instance, when the graph is altered as shown in Figure 5.3, one of the possible physical realizations is shown in Figure 5.4.

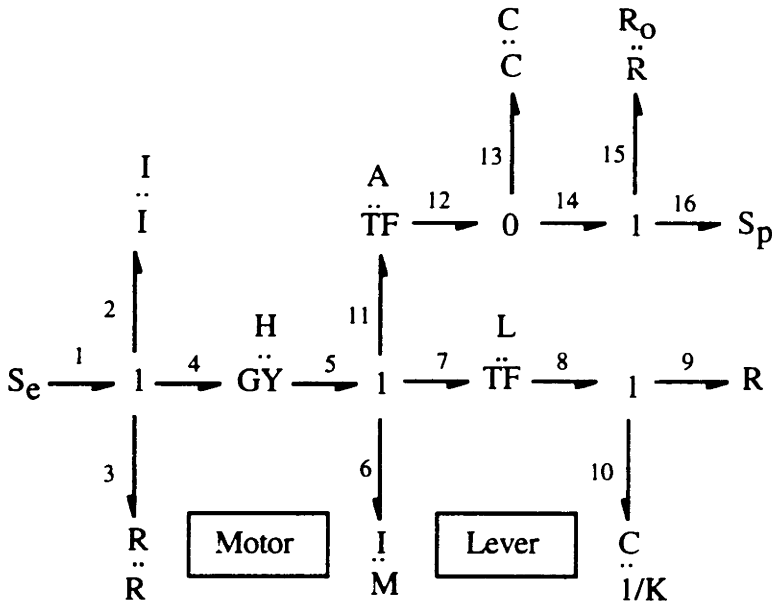


Figure 5.3 Modified bond graph of the air pump

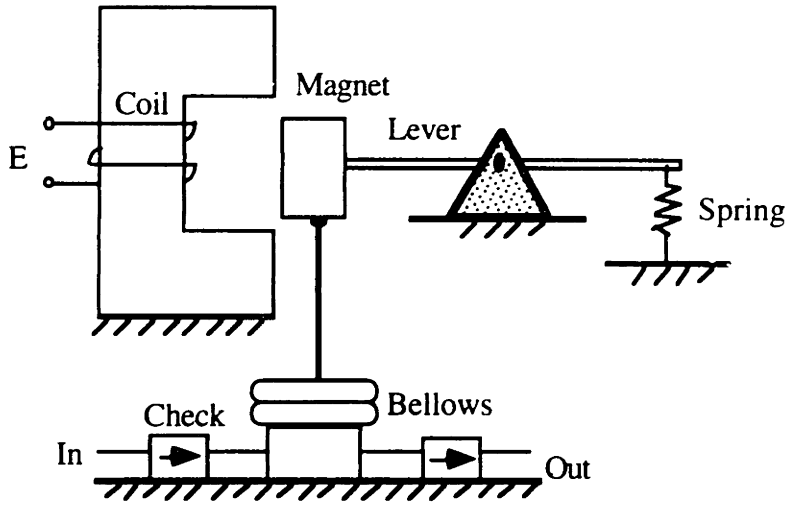


Figure 5.4 Physical system based on the modified bond graph

The mapping from bond graph representation to physical system is a one-to-many relationship. Another physical realization may be seen from the schematic shown in Figure 5.5. Owing to this one-to-many mapping property of bond graphs to the physical world, the author decided not to computerize the mapping from bond graphs to the physical world. Instead, this property is exploited as a feature for the designer to create innovative designs based on the modified bond graph representation.

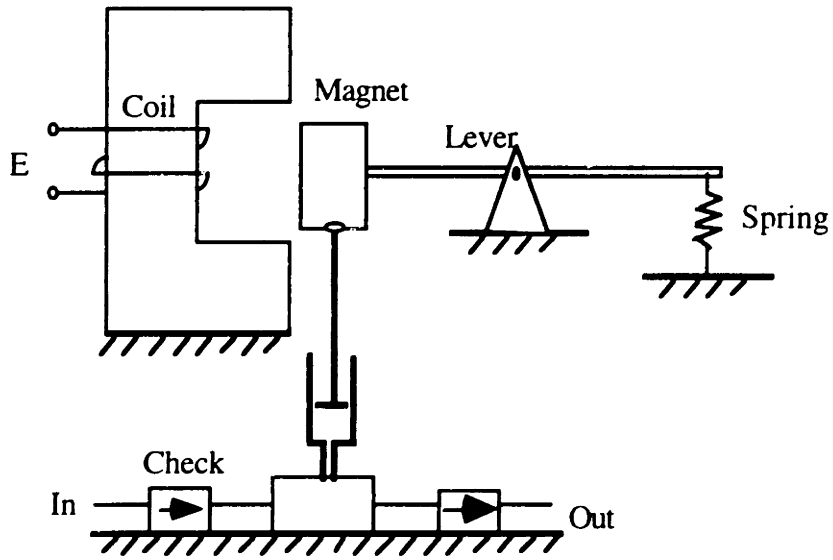


Figure 5.5 Design variant based on the same modified bond graph

One of the interesting results obtained from simulation, when the above transformation of the bond graph is done, is a higher outflow. It is assumed that this is the desired dynamic characteristic of the air pump. Note that this is achieved by changing the topology while maintaining the values of the parameters. The results from the simulation are shown in Figure 5.6 and Figure 5.7.

Thus, the use of bond graphs as a representation of dynamic systems is adequate, as it allows one to manipulate the graph to achieve the desired dynamic characteristic. To perform the operations computationally, genetic algorithms (GA) are used. The use of genetic algorithms requires a representation and a design metric. As it turns out, the representation of bond graphs is very amenable to representation of genetic algorithms utilizing a one-to-one correspondence.

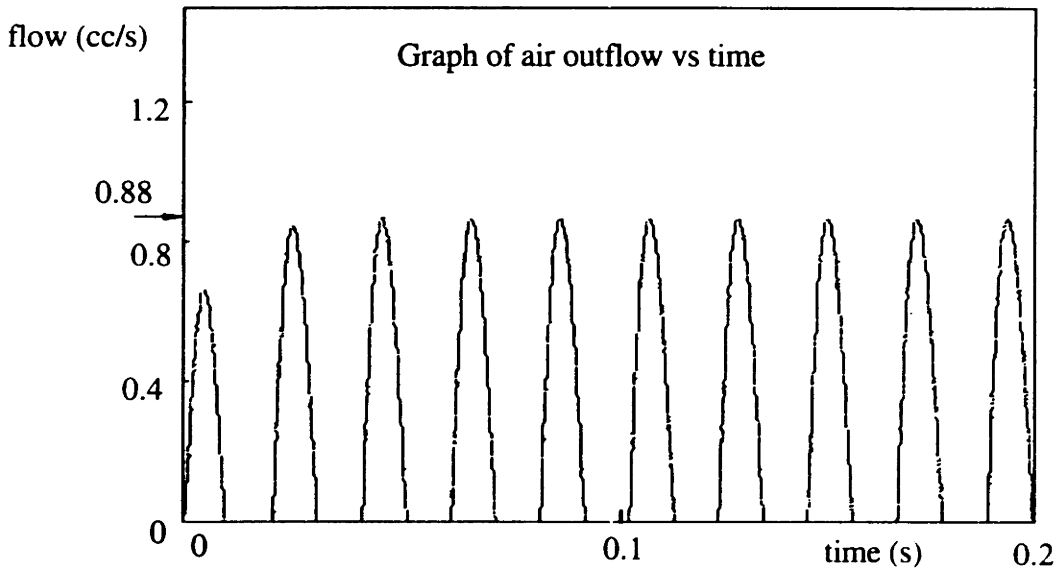


Figure 5.6 Air outflow from the original system

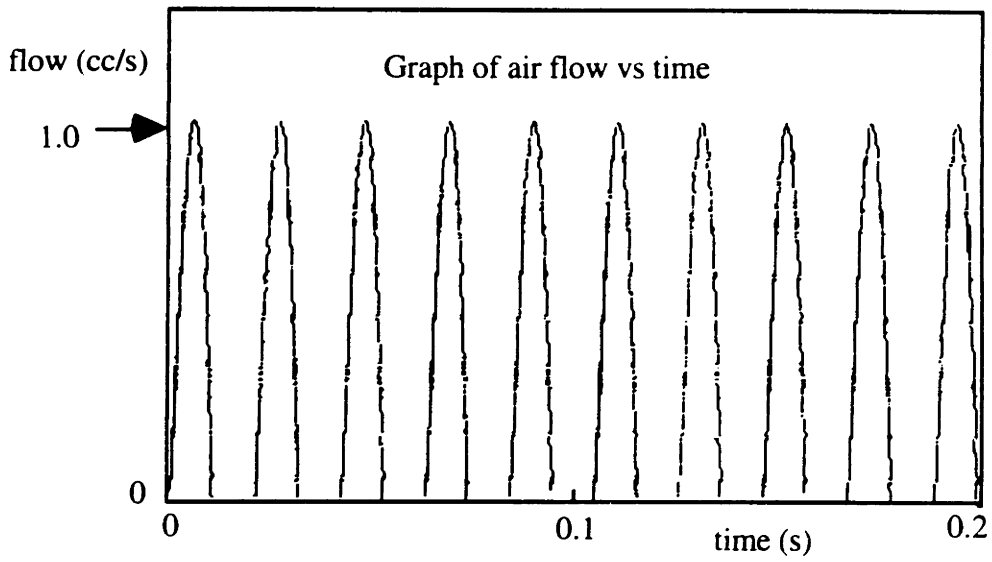


Figure 5.7 Air outflow of the reconfigured system

5.2 Genetic Algorithm Representation

Representation in genetic algorithms takes the form of strings. The bond graph topology is represented by a two-dimensional binary string, while the parameter values are represented by a binary-to-decimal string. The representation of the topology by a two-dimensional binary string is exactly the same as the incidence matrix representation into which a bond graph is uniquely mapped.

The construction of an incidence matrix is done in two steps. First, the information of the bond graph is encoded, as in schematic capture in circuits. For the bond graph shown in Figure 5.2, the result is tabulated and shown in Table 5.1.

Table 5.1 Bond graph encoding

Node	Bonds	Value
S_e	1	
SP	16	
R	3	2250 ohms
R	9	negligible
R	15	3.32×10^5 N-s /m^5
I	2	3.9 H
I	6	1.11×10^{-3}N-s^2 / m
C	13	$18.65 \times 10^{-10}$$m^5$/ N
C	10	1 / (117 Kg / m)
GY	4, 5	13 N / A ((m/s)/ V)
TF	7, 8	0.58
TF	11, 12	1.71×10^{-4} m^2
0	12, 13, 14	
1	1, 2, 3, 4	
1	5, 6, 7	
1	8, 9, 10, 11	
1	14, 15, 16	

Next, the incidence matrix is constructed. The rows represent elements of the bond graph (which correspond to the components of the physical systems), and the columns represent the bond number. The incidence matrix for the bond graph shown in Figure 5.2 is shown in Figure 5.8.

The two-dimensional binary string for the genetic algorithm is exactly the same as that of the incidence matrix. The information of the parameters encoded in Table 5.1 is encoded in the binary-to-decimal strings shown in Table 5.2.

Element	Bond Number
Se	1000000000000000
Sp	0000000000000001
I2	0100000000000000
I6	0000010000000000
C10	0000000001000000
C13	0000000000001000
R3	0010000000000000
R9	0000000010000000
R15	0000000000000010
GY	0001100000000000
TF78	0000001100000000
TF11	0000000000110000
0	0000000000011100
1	1111000000000000
1	0000111000000000
1	0000000111100000
1	0000000000000111

Figure 5.8 Incidence matrix

The summation of one's in the rows corresponds to the number of ports in the elements. The number of one's in the columns corresponds to the interconnection of the elements. The row number corresponds to the element that is represented in the table generated by the schematic capture. For example, row 3 corresponds to inertia, I2. From the incidence matrix or the two-dimensional binary string representation, I2 is a single-port element connected to a one-junction.

Table 5.2 Genetic algorithm two-dimensional binary string representation

1	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
3	0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
4	0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
5	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
6	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
7	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
8	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
9	0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
10	0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0
11	0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0
12	0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0
13	0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0
14	1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
15	0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0
16	0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0
17	0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1

5.3 Formulation

With the representation selected, the formulation takes the form of applying the operators of mutation, selection and combination in order to transform the representation from a population of randomly-generated two-dimensional binary strings to a valid bond graph with the relevant dynamic characteristics. A valid bond graph, as explained in Chapter 3, will have been subjected to several conditions, including geometric

compatibility, Newton's Law, conservation of mass, Kirchhoff's Current Law, and Kirchhoff's Voltage Law.

Further, since the goal is to generate many solutions, genetic speciation is applied as a strategy to assure a variety of solutions. For clarity of explanation, an example with fewer elements is used.

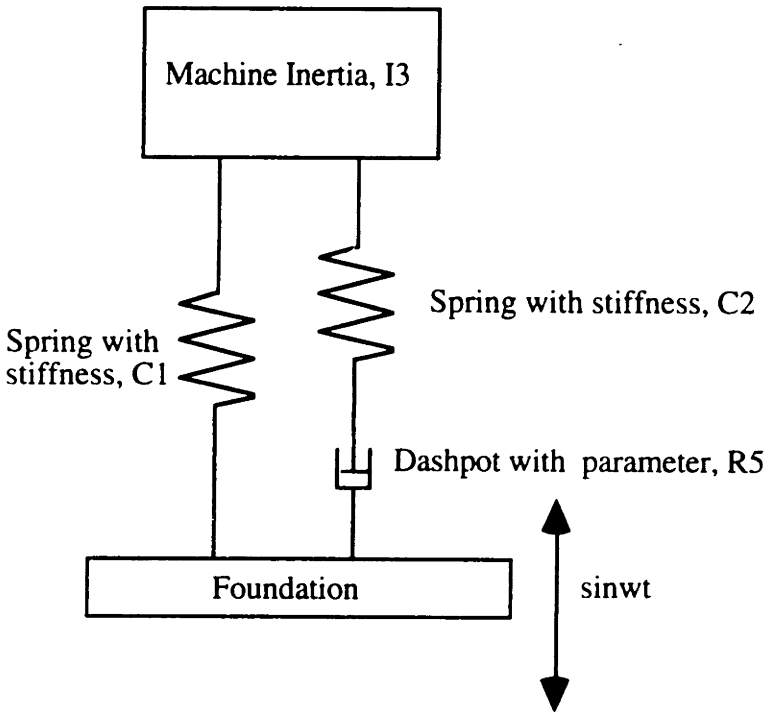


Figure 5.9 Vibration isolation system

Consider the vibration isolation system shown in Figure 5.9. In the bond graph representation shown in Figure 5.10, the dotted line shows the power flow from the vibrating foundation to the mass. The corresponding incidence matrix cum two-dimensional binary string representation is shown in Figure 5.11. However, this is the end result of the genetic algorithm run. In this case, it has resulted in a valid bond graph. What has been generated at the beginning of the run will be a population of randomly generated two-dimensional binary string representations, or "genomes" in genetic algorithm

terminology. A typical genome is shown in Figure 5.12. This genome has a fitness that is less than one, implying in this case that it does not correspond to a valid bond graph. This is shown in Figure 5.13.

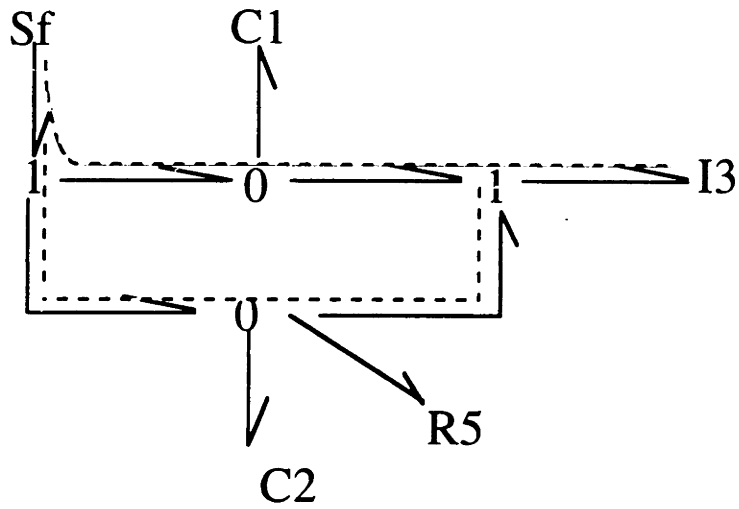


Figure 5.10 Bond graph of the vibration isolation system

```
010000000000
100000000000
000000100000
000000000000
000001000000
000000000000
000010000000
011000010000
100101000100
001100100000
000010010100
000000000000
```

Figure 5.11 Two-dimensional binary string

```
010000000000
100000000000
100000100000
000000000000
000001000000
000000000000
000010000000
011000010000
100101000100
001100100000
000010010100
000000000000
```

Figure 5.12 Two-dimensional binary string that has a fitness less than one

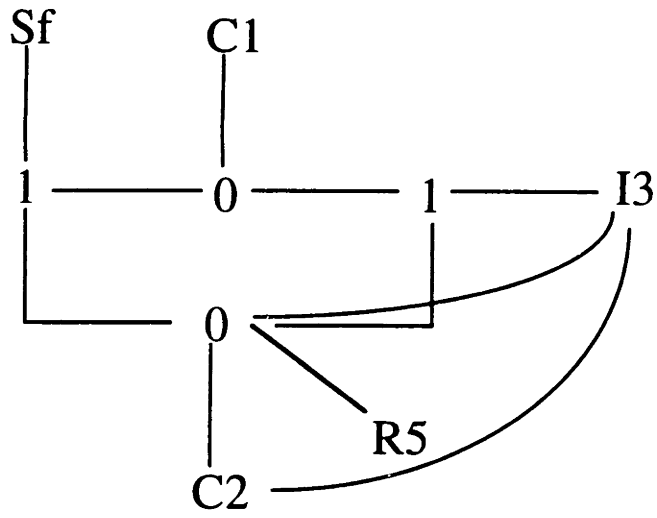


Figure 5.13 Invalid bond graph

The sections to follow will provide the formulation that transforms a somewhat meaningless two-dimensional binary string that has been randomly generated, to a bond graph that exhibits the desired dynamic characteristics.

5.3.1 Fitness Function

The fitness function for topology is described here, as it is consistent throughout the generation of valid graphs. The fitness function of the dynamic characteristics is

different for different case studies (since the requirement of an accelerometer cannot be the same as that of an aquarium pump). This is described in Chapter 7.

The fitness function has the objective of achieving a valid bond graph. A score of one indicates that a valid graph has been obtained. In the first portion, the number of ports and interconnections is checked. A single-port element should have a single port; only two elements can be connected to one another if they are to be connected at all. From Figure 5.14, the number of ones in the first row is good, as C1 is a single-port element. This can be expressed succinctly by the following mathematical equation:

$$\text{row fitness, } f1 = (\sum f_i)/n \qquad (5.1)$$

where i = 1, 2, ..., n

and n = number of rows.

$f_i = 1$ for $\sum_j = 0, 1$, where j = number of columns

$1/\sum j$ otherwise.

$\sum j = 0$ implies that the element has no port, which means that it cannot be connected, and therefore is not featured in the representation.

C1	010000000000
C2	100000000000
I3	100000100000
I4	001000010000
R5	110011000000
R6	111000000000
SF	000001000000
0	001111000000
0	000001111000
1	000000000000
1	000000011100
1	111110000000

Figure 5.14 Calculation of fitness value

A similar procedure is followed for the column fitness. Calculating the fitness of the columns is slightly more involved, as the terminal elements such as inertias, capacitors and resistors cannot be connected to one another; the junction structures such as transformers, one-junction and zero-junction can be connected one to another and with the terminal elements. The calculation of the column fitness is therefore adjusted to take this into account. In the first calculation, the column fitness of the single-port elements is made to ensure that no terminal element is connected to another terminal element. Row fitness is done in a similar way. In the second calculation, the entire column is considered to ensure that only two elements are connected, if there is to be a connection at all. Again, the calculation takes a similar form to that of the row fitness. The three fitnesses are multiplied by a profile to make them sum to one if the topology results in a valid graph.

The fitness is then modified by a factor α, which takes into account causality. As explained in Chapter 2, integrative causality is the preferred causality, as it implies no physically impossible solutions and/or fully observable and fully controllable systems. Though the issue of observability and controllability is not dealt with in this thesis, these are indeed good characteristics for a dynamic system to have.

α is the proportion of storage elements with integrative causality, to the number of storage elements. The desired value of α is 1. At this stage, if the topology fitness is 1, the resultant bond graph will be a valid bond graph with integrative causality for all the storage elements.

5.3.2 Selection and Crossover

Consider the two genomes shown in Figure 5.15. Their respective fitness scores are shown. It is assumed that their fitnesses put them in the elite class that is selected by the roulette-wheel method of procreation. A random crossover point is generated. The top portion of the first parent merges with the bottom portion of the second parent to form a child, and the second child is formed from the combination of the bottom portion of the first parent with the top portion of the second parent.

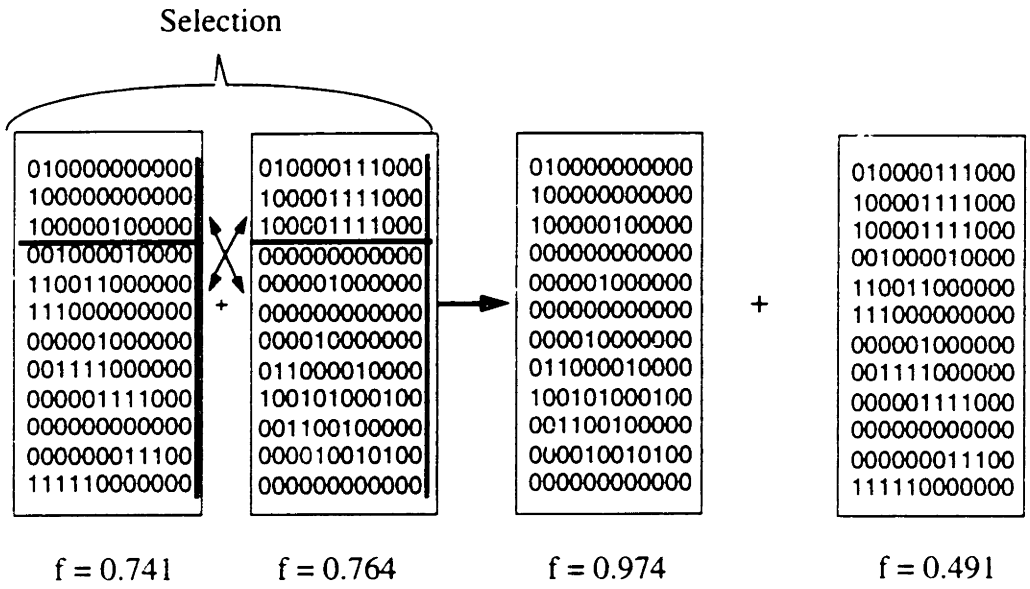


Figure 5.15 Operation of combination using crossover

The first child is selected for the next generation, while the second child is discarded if it ranks near the bottom of the population. From this illustration, it is noted that crossover does not necessarily create superior offspring; and in the instance above, one offspring is less superior than the parents.

For the purpose of clarity, the development of the superior offspring will be followed, and the effect of the operation of mutation examined.

5.3.3 Mutation

The operation of mutation involves the toggling of the value in one of the positions of the two-dimensional binary string. This position is selected randomly. As can be seen in Figure 5.16, the value of the first column of the third row is toggled. This corresponds to a dramatic change in the bond graph representation, as shown in Figure 5.17.

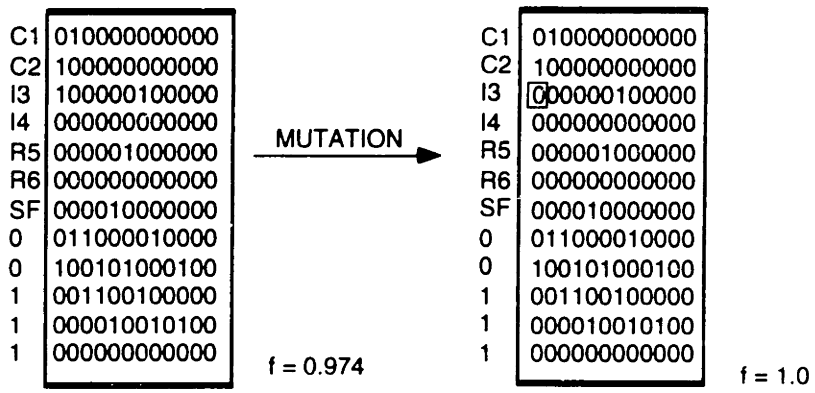


Figure 5.16 Operation of mutation

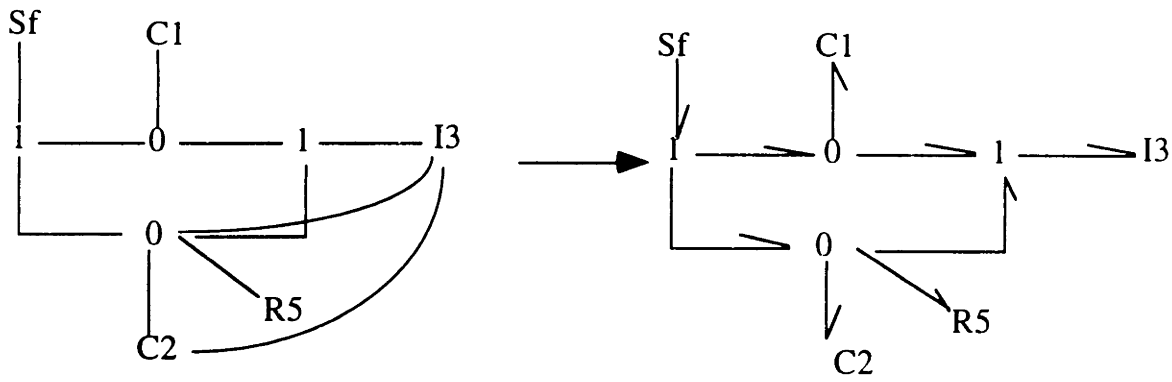


Figure 5.17 Effect of mutation on bond graph

Thus, the operations of crossover, selection and mutation have profound effects on the bond graphs. From an invalid graph, which corresponds to an inept formulation of the problem of vibration isolation of the machine from the vibrating foundation, the resultant graph is one design variant capable of satisfying the requirements of dynamic characteristics. This is due to the mathematical formulation that the bond graph embodies. This formulation relates the output (in this case, the velocity of inertia, I3) to the input (the velocity of the flow source).

The complete assignment of power and causality follows. The graph is then converted into the state space representation, from which the dynamic characteristics are determined. At that stage, the computation proceeds as in parametric design. While the topology is maintained, the values of the parameters are varied. The dynamic characteristic is calculated and compared with the desired value. When the desired value is reached, a solution is obtained.

This thesis, however, is about the generation of design variants, and not about one design solution. Therefore, a strategy must be used to force the generation of variants. This strategy is genetic speciation, described in Chapter 2.

5.3.4 Genetic Speciation

Consider the left genome (Figure 5.18) which has been selected for comparison. The fitness of the rest of the population is then derated according to the triangular function described in Chapter 2. Although the formula calls for the evaluation of the sharing function for the entire population, the effects of the sharing function on the candidates can be seen even by the simple calculation of the distance between the candidate and the selected genome. The first candidate (the second genome) has a sharing function close to

one; that is, it is almost identical to the first genome; therefore, the derated function is mu less than that of the third genome. The third genome has a sharing function value much less than one, which implies that it is very different from the first. The derated fitness is therefore higher, and the third genome stand a higher chance of being selected for procreation in the next generation.

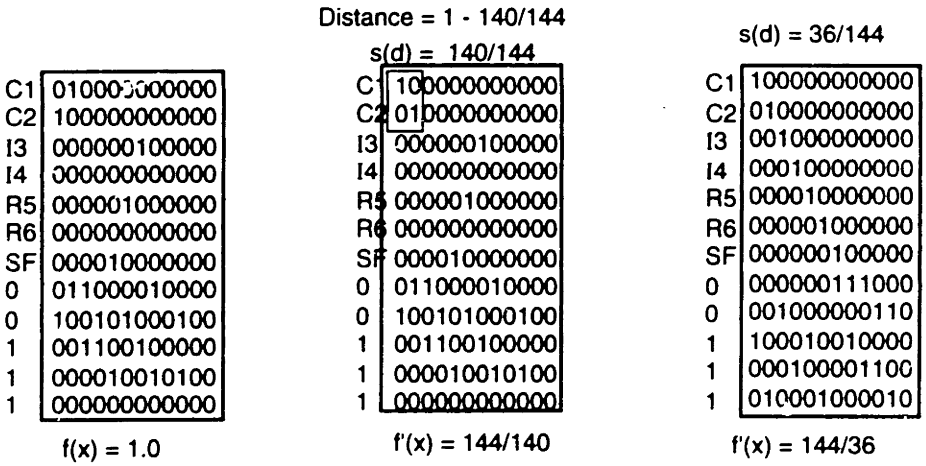


Figure 5.18 Strategy of speciation

The speciation strategy seen in bond graph terms will make it easier to appreciate.

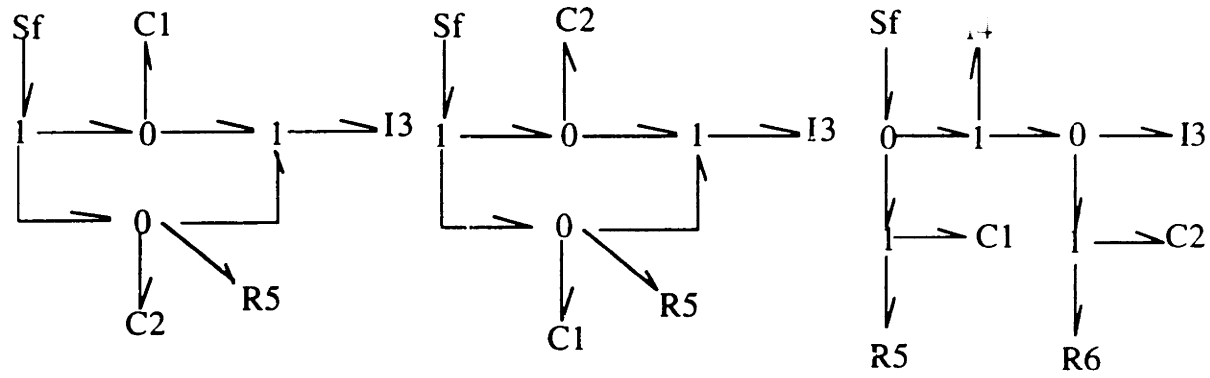


Figure 5.19 Speciation applied to bond graphs

The first bond graph In Figure 5.19 corresponds to the first genome in Figure 5.18: that is, the selected graph for comparison with the candidate graphs. The second graph, which corresponds to the second genome, differs from the first by the position of the springs. Thus, spring C1 is in series with the dash pot in the second case, while spring C2 is in series with the dash pot in the first.

In the third bond graph, the topology is very different. Indeed, there are more elements than in the first bond graph, as inertia I4 has been added to the graph. Since the third bond graph is preferred over the second bond graph for procreation, the strategy results in the generation of design variants.

Chapter 6

Implementation

6.0 Overview

Altogether there are three programs written: "Topology Generator," "Performance Generator," and "Variant." Topology Generator performs the reconfiguration of topology in the bond graphs. Performance Generator uses a topology, and varies the values of the parameter to give the desired performance. Variant is a combination of both Topology Generator and Performance Generator.

Section 6.1 gives a description of the developed program, Variant. Section 6.2 outlines the inputs required. Section 6.3 describes the component software modules of Variant.

6.1 Introduction to Variant

Variant is written in an object-oriented language, C++. It runs on the Silicon Graphics Iris R4000 computer. There are three main modules of Variant. The first is the genetic algorithms that encapsulate the operators of transformation. The second is the bond graph program that formulates the problem for the program. The third is the MATLAB™

program that performs the evaluation of the dynamic characteristics of bond graphs. The organization of the program modules is shown in Figure 6.1.

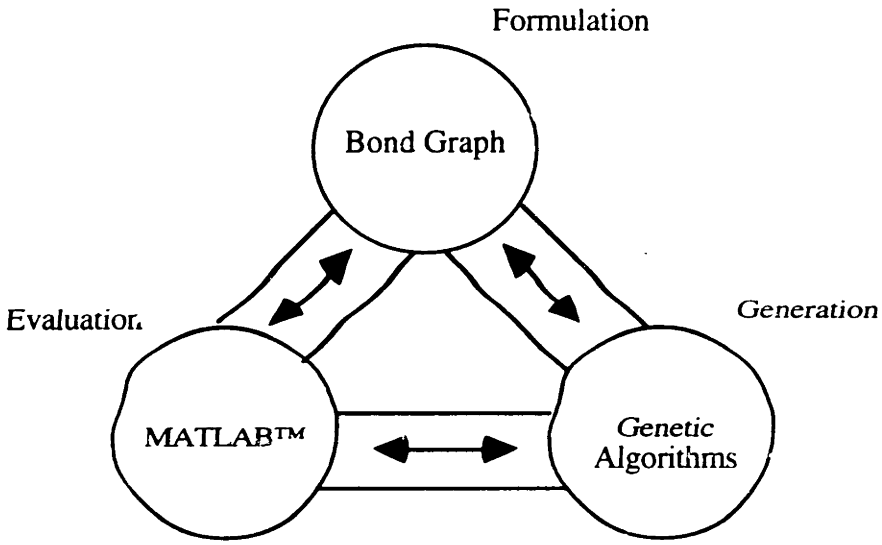


Figure 6.1 Organization of the software modules

Two modules, genetic algorithms and bond graphs, constitute Topology Generator. The objective is to find valid bond graph structures with the required input and output elements and a reasonable intervening structure. The designer will have to decide what are the criteria for a reasonable structure. In the case studies, the maximum order of the system is used as a criterion for the complexity of the systems with which the designer is willing to contend.

Performance Generator is essentially made up of MATLAB™ and the genetic algorithm. Variant consists of Topology Generator and Performance Generator, but the inclusion of these two generators requires the third link between the bond graph and MATLAB™ to be established. MATLAB™ needs to be interfaced with the bond graph module in order to understand the representation which has been evaluated for formulation.

(The Performance Generator assumes that the bond graph given to the program is a valid bond graph.)

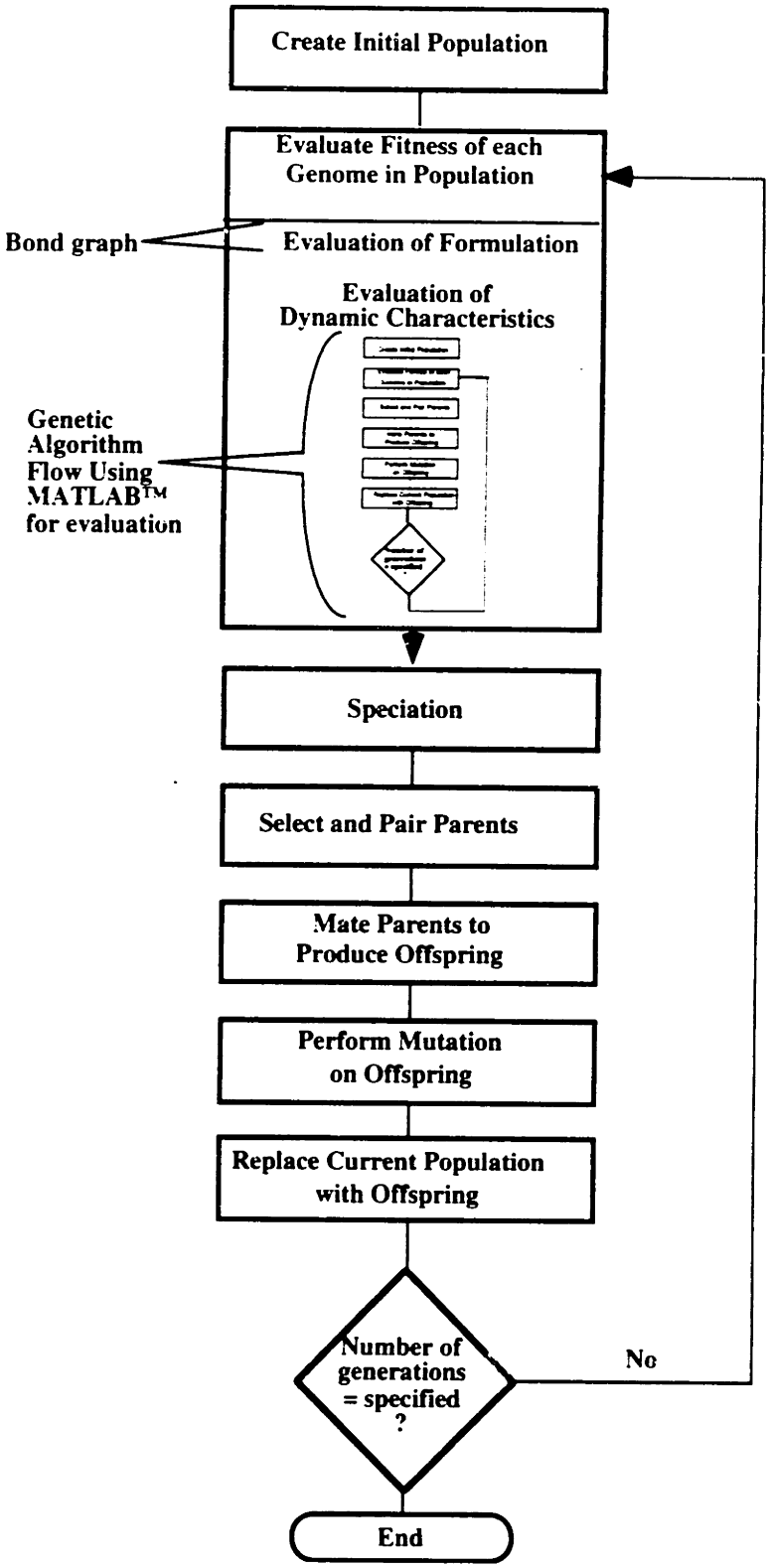


Figure 6.2 Flow of the program

The flow of the program is shown in Figure 6.2. Genetic algorithms generate various potential designs, which are evaluated for their formulation based on the construction of the equivalent bond graph structure. Once a valid bond graph is found, the MATLAB™ module is used to evaluate the dynamic characteristics by varying the values of the parameters. This operation is accomplished by use of the genetic algorithm. Once the pre-specified number of generations is reached, control is returned to the outer genetic algorithm loop. This loop exercises the speciation strategy to generate as many different topologies as possible.

6.2 Inputs to the Program

The designer specifies the following inputs:

1. Maximum number of each component.
2. Range of values for each parameter that characterized the component.
3. Input (s) to the dynamic system.
4. Output(s) from the dynamic system.
5. Desired performance of the dynamic system.

In addition, for the genetic algorithms, the designer will have to specify the following inputs:

1. Number of generations to be performed for the Topology Generator.
2. Size of the initial population.
3. Probability of crossover.
4. Probability of mutation.

Another set of similar input is required for the nested genetic algorithm that generates the parameter values.

The input to the program also includes the designer-specified output of interest. The range of acceptable values for the dynamic characteristic also needs to be specified.

6.3 Program Modules

6.3.1 The Genetic Algorithms (GAs)

There are primarily two classes in the genetic library: the genome, and the genetic algorithm. Each genome object represents a single solution to the problem. The genetic algorithm object defines how the evolution should take place. In addition to these two objects, an objective function that is relevant to the problem must be constructed. The objective function tells the GA how well each genome "performs."

In Variant, the genetic algorithm used for varying the topology is the steady-state genetic algorithm.

In using a genetic algorithm to solve an optimization problem, the problem must be represented in a single-data structure. The genetic algorithm operates on these structures to evolve the best solution. In the GA library, the basic structure is called a GAGenome. There are three classes of structures derived directly from the GAGenome: GAListGenome, GATreeGenome and GAArrayGenome. The derived genome classes descend not only from the GAGenome class, but also from a data-structure class as reflected in their names. In variant, the derived genome classes used are a 2-dimensional binary string and a binary-to-decimal string.

Each genome has three primary operators: initialization, mutation, and crossover. The initial population can be biased with the initialization operator. The mutation and crossover are defined specifically for each particular problem, according to the representation. Essentially, the initialization operator determines how the genome is initialized. The mutation operator defines a method for mutating the genome. The crossover operator defines the procedure for generating a child from two parent genomes. It is used in conjunction with a crossover site object.

In general, the genetic algorithm does not need to know about the contents of the data structures on which it is operating. The library reflects this generality. Mixing and matching genome types with genetic algorithms is possible; this feature is exploited in Variant by using a two-stage genome. The second stage of the binary string that carries the parameter-values information is used only when the bond graph represented by the two-dimensional genome is valid.

The genome's member functions perform various roles, including cloning the population to create the future population, initializing genomes to restart a run, and crossing genomes to generate children and mutate genomes.

The genetic algorithm contains the selection method, statistics, replacement strategy, and parameters for running the algorithm. In Variant, the selection method used is the roulette-wheel method. The replacement strategy is the elitist replacement with cmult set at 2. That is, there will be copies of the best member in the next population. The parameters used in variant change according to the purpose of the run, but in general, it is set at 20,000 generations with a population size of 30. Probability of mutation is set at 0.00132, and probability of crossover is set at 0.8. These rates were determined, after

some trials, to be satisfactory for convergence. Overlapping (steady-state) genetic algorithms are used in Variant.

The population object is a container for genomes. Each population object has a scaling-scheme object associated with it. The scaling-scheme object converts the objective score of each genome to a fitness score that the genetic algorithm uses for selection. In variant, the scaling scheme adopted is the linear- proportion scaling scheme.

It is important to note the distinction between fitness and objective scores. The objective score is the value returned by the objective function. The fitness score is the (possibly scaled) value used to determine fitness for mating. They are not necessarily the same. The genetic algorithm uses the fitness scores, not the objective scores, to make the selection.

6.3.2 The Bond Graph Module

Variant essentially utilizes a graph description, or a degenerate form of graph description specifying the maximum number of each element and the interconnection among the elements. The program then determines the formulation of a valid bond graph. The complete formulation of the bond graph is verified by the eigenvalue calculation. The successful determination of the eigenvalues indicates that the minimally-connected graph has been properly assigned its causality and power.

The bond graph program can be broken down into a number of classes, including ElementList, fifo and gnode class. The ElementList class performs several class functions, such as automatic assignment of flows and power directions, vital in determining the

causality of bond graphs. The fifo class keeps track of the stack operations used extensively in the determination of the eigenvalues. The gnode class is used in the drawing operations for output purposes.

There are several junctures in the program where the fitness of the representation of the bond graph is evaluated. This is done sequentially, so that if a certain criterion such as integral causality is not met, the graph is assigned a fitness score and the program flow is directed back to the genetic algorithm module.

The bond graph program is based on the algorithm of partitioning matrices. The specific problem addressed by this program is this: given a bond graph composed of elements from the basic set {C, I, R, Se, Sf, TF, GY, 0, 1}, find a method of generating state space equations of the form:

$$\begin{aligned} \dot{X} &= AX + BU \\ Y &= CX + DU \end{aligned} \qquad (6.1)$$

The desired outputs from the bond graph module are the state-space matrices: A, B, C and D matrices. These matrices in turn form the input to the MATLAB™ module of the program.

6.3.3 The MATLAB™ Module

This is a commercial program from Mathworks© that integrates matrix computation, numerical analysis, signal processing, data analysis, and graphics in an easy-to-use environment.

In Variant, MATLAB™ is used as an engine, that is, not in an interactive mode. The incorporation of MATLAB™ as an engine opens Variant to a wide range of application-specific solutions called toolboxes. Toolboxes are libraries of MATLAB™ functions that customize MATLAB™ for solving particular classes of problems. In the case study, MATLAB™ is used to solve dynamic problems; therefore, the control toolbox that determines dynamic characteristics is utilized. Transmissibility, phase margin, gain margin, and other relevant quantities may be calculated using MATLAB™.

Chapter 7

Case Studies

7.0 Overview

This chapter records the case studies that were done to verify the developed method of design generation described in the previous chapters. The first case study involves the design of a vibration isolation system. This case illustrates the use of bond graphs to represent adequately the design and results obtained when the operators of design were applied to the design. This case also illustrates the choice of inputs: the desired dynamic characteristic (transmissibility); a maximal set of elements, indicating the maximum order of the system with which the designer is willing to contend; the nominal values of the parameters; and the range of values of the same.

The second case study is the design of an accelerometer to satisfy the dynamic characteristics of phase margin and gain margin. This case illustrates the wide range of dynamic characteristics with which the program can deal.

The third case study is the design of a pump system. This example illustrates the versatility of the program in handling design which involves inter-domain subsystems: electrical motors, mechanical mechanisms, and fluid flow elements. The program

essentially generated a number of topologies that are possible solutions to the problem of finding possible interconnections between the output of fluid flow and the input of electrical energy.

7.1 Case Study 1: Vibration Isolation Systems

7.1.1 Introduction

Vibration isolation is a vibration control technique, wherein an isolator having suitable characteristics is inserted between the vibratory source and the system requiring protection, to reduce the level of transmitted vibration. Insertion of the isolator generally creates a resonance in the frequency response of the vibration isolation system, above which the level of transmitted vibration decreases with an increase in excitation frequency. Vibration isolation systems can be generally categorized as linear or nonlinear, depending on whether or not their vibration response is described by linear differential equations with constant coefficients. They can be further categorized as active or passive, depending on whether or not external power is required for the isolator to be operational. In this research, only passive and linear-vibration isolation systems are considered.

The vibration isolation problem is illustrated in Figure 7.1.

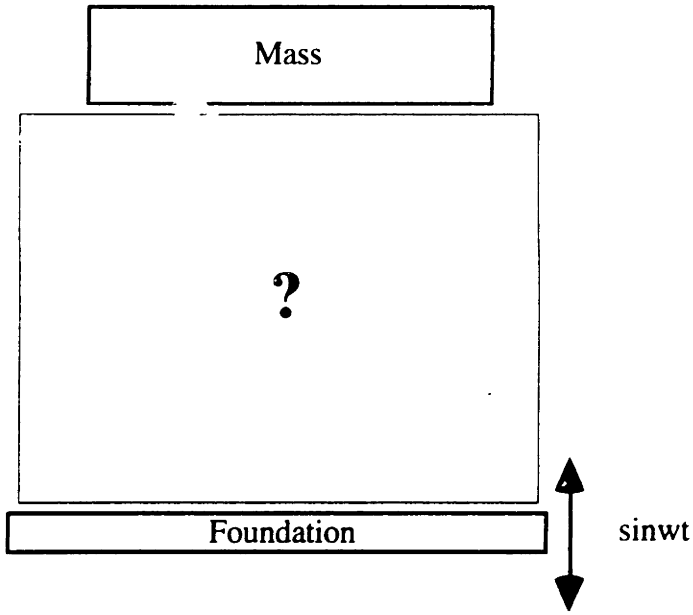


Figure 7.1 Vibration isolation problem

Here the mass can be a precision measuring machine on the shopfloor, or a control panel for imaging on the printing press. Since the resolution required in both applications is very demanding, the mass must be isolated from the vibrating foundation to a high degree. In this research, the machine is isolated to a hundredth of a percent of the velocity of the vibrating foundation.

7.1.2 Basic Assumptions

Lumped-parameter mathematical models are frequently used in the analysis and design of vibration isolation systems, as well as in the interpretation of characteristics of vibrating mechanical systems. Although the lumping of isolation system properties into rigid mass, ideal stiffness, and damping elements represents a simplification of reality, the usefulness of vibration response characteristics based on this simplification has long been established. Consequently, to avoid unnecessary complexity, the isolation system is

considered to be comprised of a rigid mass supported on a rigid foundation by a single isolator undergoing unidirectional vibration in response to harmonic vibration excitation of angular frequency ω. The schematic diagram of an idealized vibration isolation system is illustrated in Figure 7.2, where the excitation is represented by a harmonic displacement a(t) = sinωt of the rigid foundation. For vibration excitation of the foundation, the purpose of the isolator is to reduce the magnitude of vibratory velocity $\dot{x}(t)$ transmitted to the isolated mass.

Only viscous damping is considered in this case study, as it is considered to be linear. This is a good approximation at low speed, where the flow through the orifice is laminar. The energy dissipated per cycle by a viscous damper experiencing a harmonic relative displacement z=zosinwt is dependent on the frequency and the amplitude of vibration.

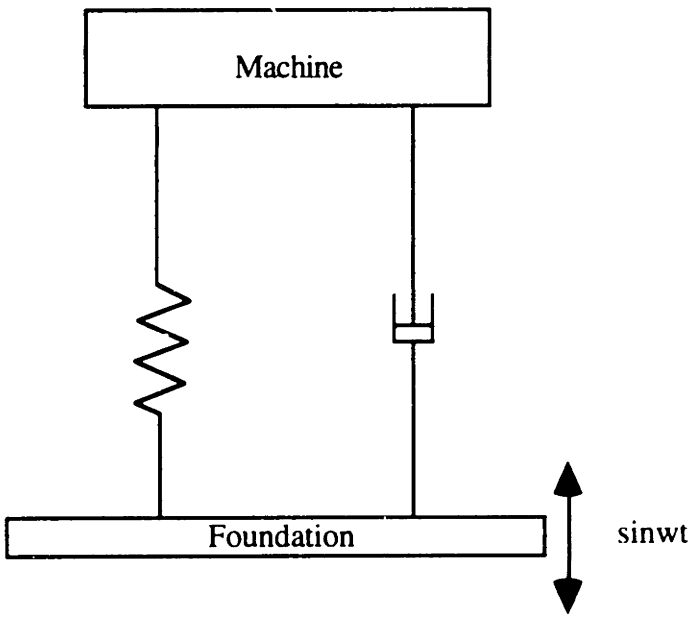


Figure 7.2 Vibration isolation for a machine

7.1.3 Isolation System Response Parameters

The primary responses of interest for vibration excitation of the foundation are the absolute displacement of the mass x, or its derivatives, and the relative displacement $\delta = (x - a)$. Absolute response parameters such as velocity $\dot{x}$, acceleration $\ddot{x}$, or force F_T are important with regard to fragility and structural integrity properties of the isolated body or foundation, whereas the relative displacement δ across the isolator is important in the determination of isolator strength and isolation system clearance requirements. For this case study, only the absolute velocity was taken into account, as the other quantities can be derived from it.

In particular, transmissibility is used as the dynamic characteristic of interest. Transmissibility is the ratio of the output velocity to that of the input.

7.1.4 Inputs to the Computer Program

The inputs to the system are as follows:

1. Maximum number of storage elements: two masses and two springs.
2. Maximum number of dissipative elements: two
3. Range of parameter values: from one-tenth the nominal to ten times the nominal
4. Nominal values:
   - masses: I3 = 1000kg and I4 = 100kg
   - springs: C1 = 5000N/m and C2 = 2000N/m
   - dissipative elements: R5 = 600N/m/s and R6 = 200N/m/s.
5. Input quantity: velocity of foundation = sinwt
6. Output quantity: velocity of mass, I3

Parameters for the genetic algorithms are set as follows:

For the first stage:

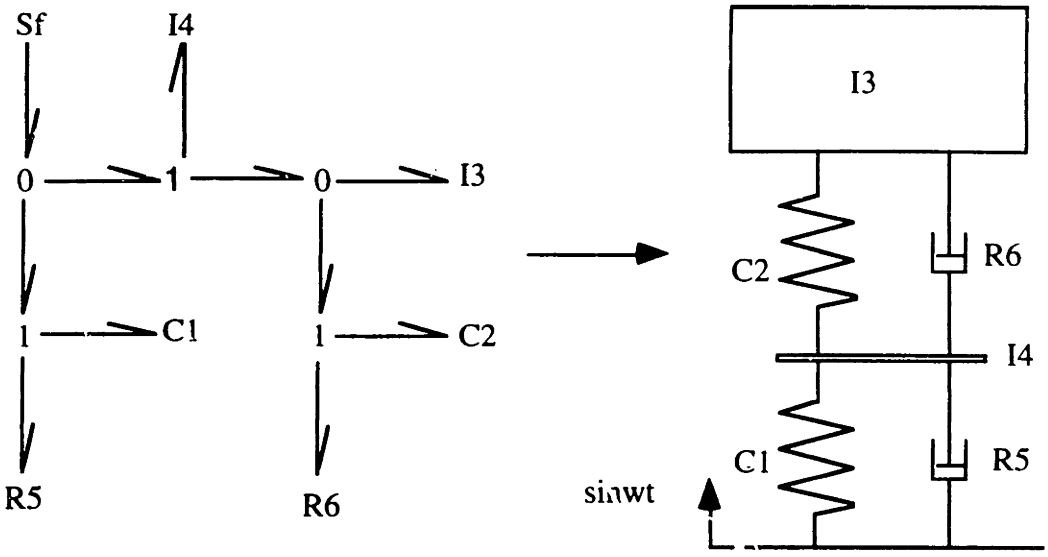
1.	Number of generations:	20000
2.	Size of population:	80
3.	Probability of crossover:	0.8
4.	Probability of mutation:	0.013
5.	Proportion for replacement:	0.9
6.	Sharing function used:	triangular

For the second stage:

1.	Number of generations:	200
2.	Size of population:	10
3.	Probability of crossover:	0.8
4.	Probability of mutation:	0.013

7.1.5 Results from the Computer Program

The results from the computer program are shown in the following figures.



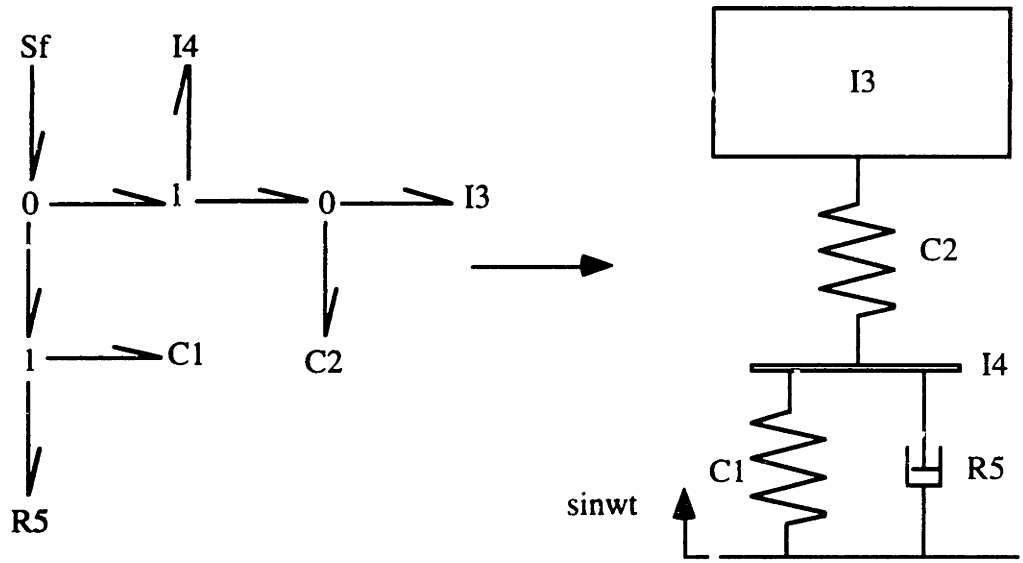
Values of Parameters:

C1	C2	R5	R6	I4	I3
30782N/m	11614N/m	60N/m/s	1728N/m/s	207kg	10000kg

Transmissibility:

4.05exp(-8)

Figure 7.3 Vibration isolation design variant 1



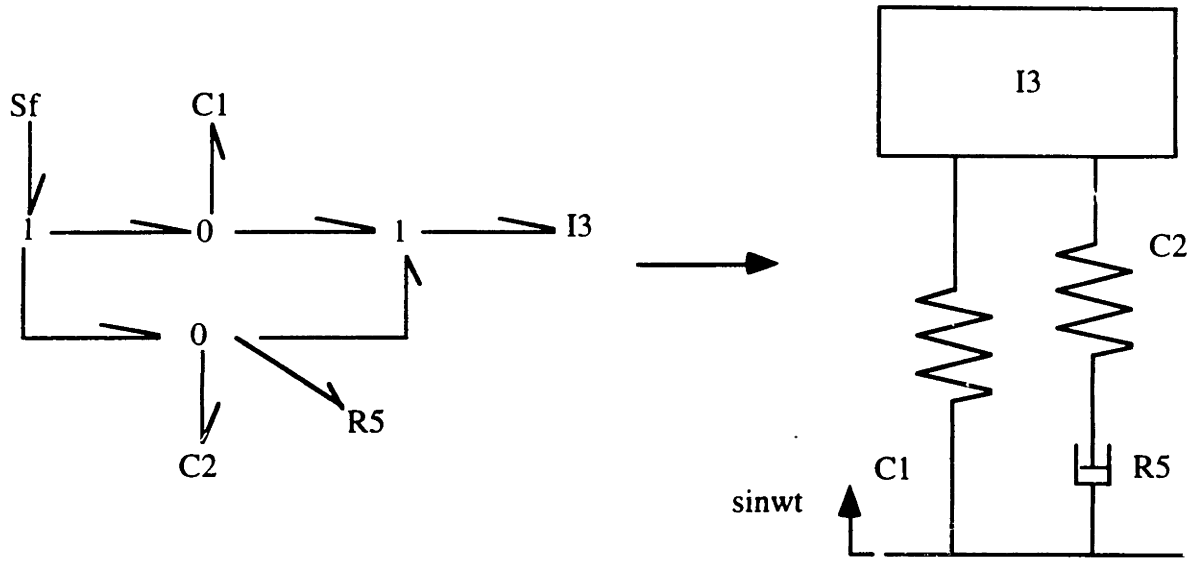
Values of Parameters:

C1	C2	R5	I4	I3
41459N/m	200N/m	340N/m/s	165kg	10000kg

Transmissibility:

7.80exp(-8)

Figure 7.4 Vibration isolation design variant 2



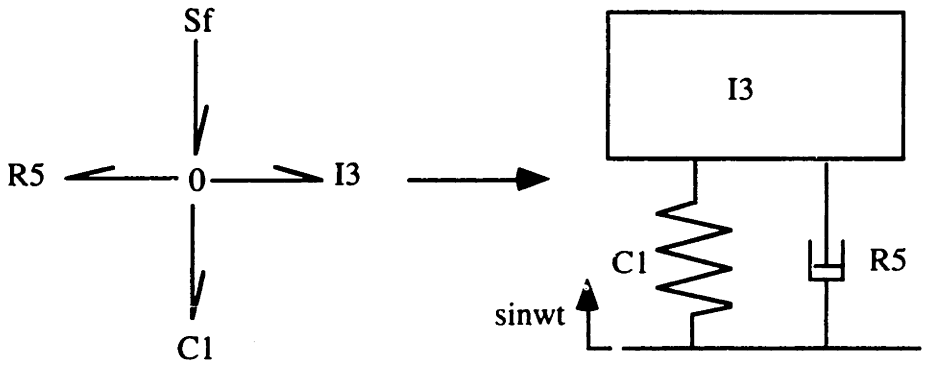
Values of Parameters:

C1	C2	R5	I3
44177N/m	666N/m	5324.5N/m/s	10000kg

Transmissibility:

3.172exp(-5)

Figure 7.5 Vibration isolation design variant 3



Values of Parameters:

C1	R5	I3
15970N/m	600N/m/s	10000kg

Transmissibility:

1.1268exp(-5)

Figure 7.6 Vibration isolation design variant 4

From the design variants, the designer can develop insight into where the absorbing material can best be placed. Further, if the number of components is an issue, the designer may opt for variant 4 to satisfy the dynamic requirements.

Figure 7.7 shows the results of a typical best-of-generation run from the genetic algorithm.

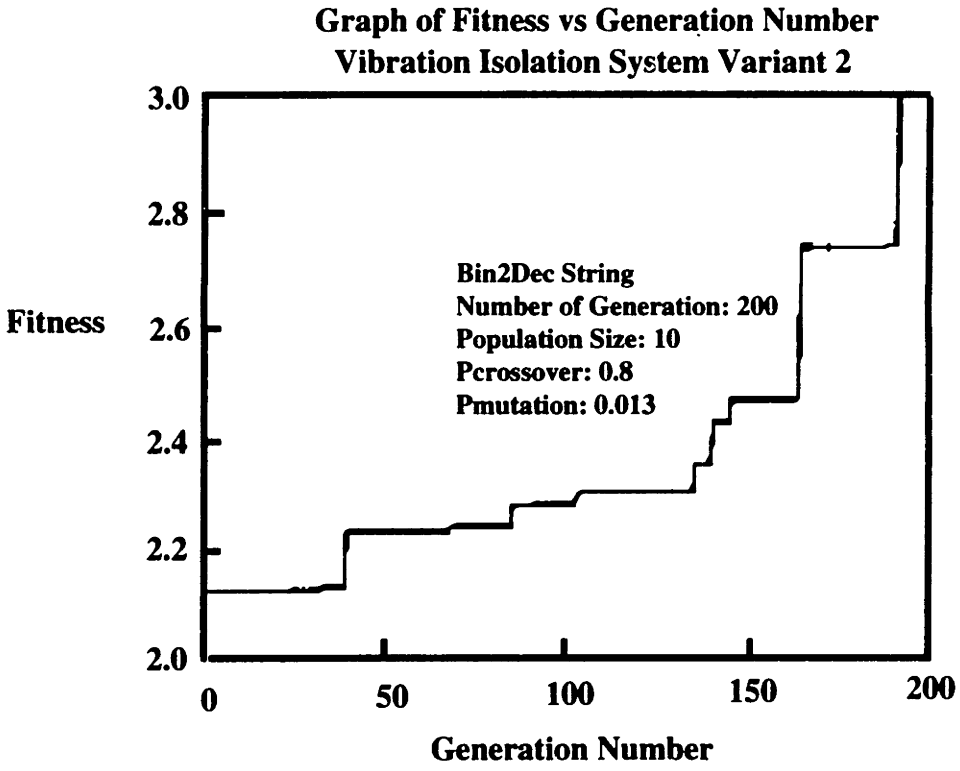


Figure 7.7 Graph of Fitness vs Generation Number

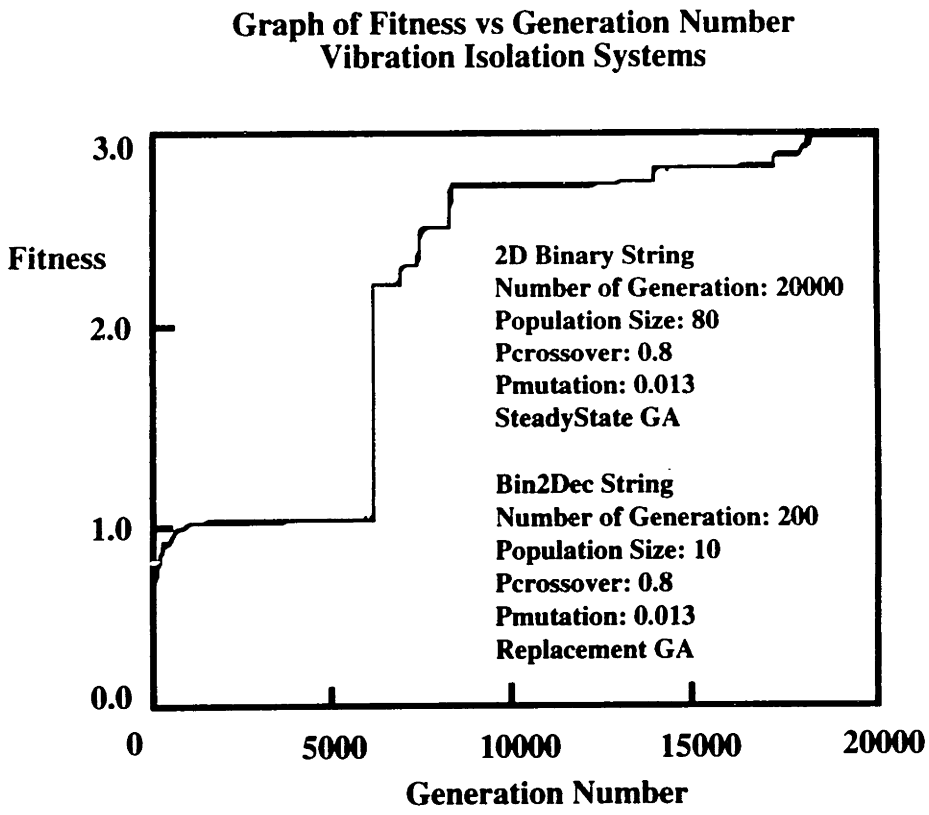


Figure 7.8 Graph of Fitness vs Generation Number for entire run

The results are from the nested genetic algorithm which varies the values of the parameters. It shows that the fitness value of the population increases as the number of generations increases. It continued to increase, and was stopped at generation number 200, as this met the termination criteria. Any graph structure with fitness value above 3 would have satisfied the dynamic characteristic requirement.

In Figure 7.8, the fitness value of the best-of-generation for the entire run is shown. Bond graphs with fitness values above 1.0 are valid graphs. Bond graphs with fitness values of 3.0 have satisfied the dynamic characteristics.

7.2 Case Study 2: The Design of an Accelerometer

7.2.1 Introduction

An accelerometer is a device that measures acceleration. Typically, the accelerometer produces a displacement proportional to an acceleration of the device reference frame.

One of the more common accelerometers is the suspended-mass type, characterized by their small size (solid-state devices are about as big as a transistor), and the ability to respond to dc (static) accelerations. Frequency response ranges from dc to about 5kHz, depending on the particular type. High g-range units generally have higher frequency response than their low-range equivalents. A simple model of such a device is mass-suspended on the end of a cantilever beam (as shown in Figure 7.9), but actual devices are more complicated.

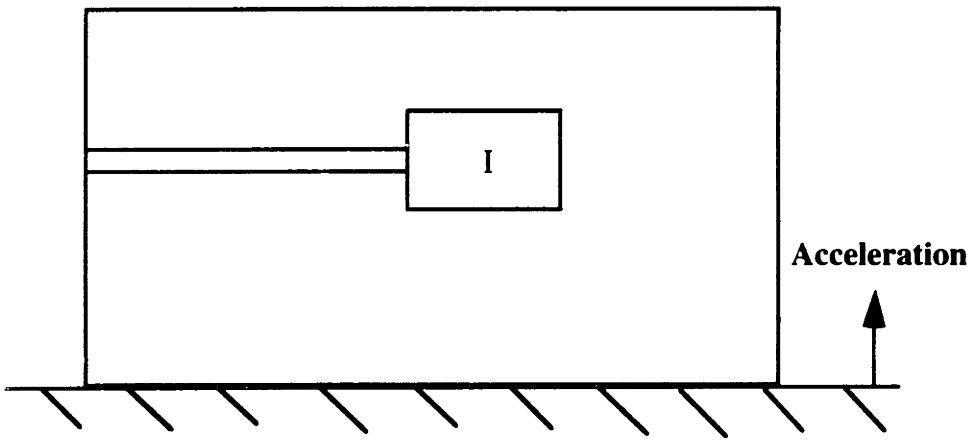


Figure 7.9 Schematic of an accelerometer

7.2.2 Some Results

One of the bond graphs generated is shown in Figure 7.10.

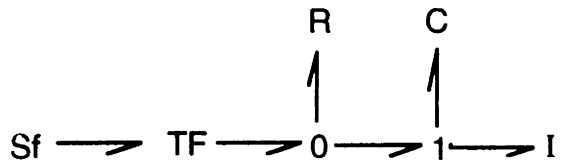


Figure 7.10 Design variant for the accelerometer

The desired dynamic characteristics include a phase margin between 30° to 60°, and gain margin equal to or greater than two.

The results of the generation are as follows:

mass of I = 5g

stiffness of cantilever rod = 95N/m

dissipative constant = 0.005N/m/s

transformer modulus = 0.03.

With the above parameter values, the phase margin obtained is 40° and the gain margin is 2.

7.3 Case Study 3: Air Pump System

7.3.1 Introduction

This is an application of the Variant program to generate different pump systems. The original design is shown in Figure 7.11. This is an example taken from [Martens 70].

The functional requirement is to supply air. The input to the system is a line voltage. The objective is to generate different topologies of bond graphs with a natural frequency of 60Hz, as this is the line frequency.

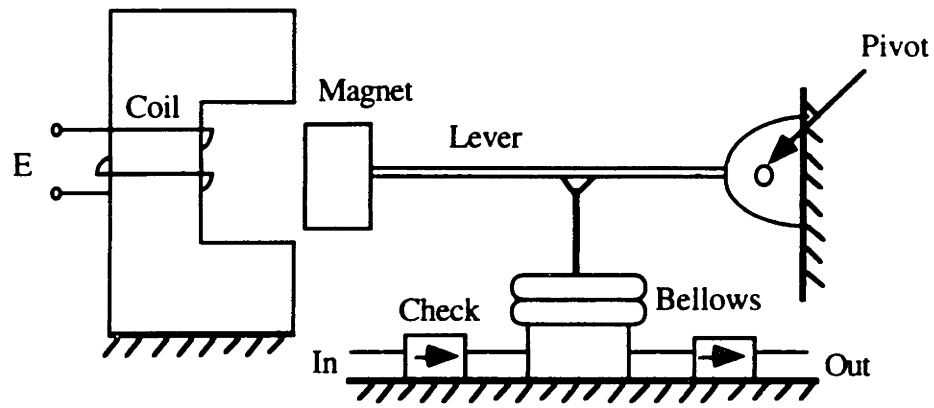


Figure 7.11 Schematic of air pump

The bond graph of the above system is shown in Figure 7.12.

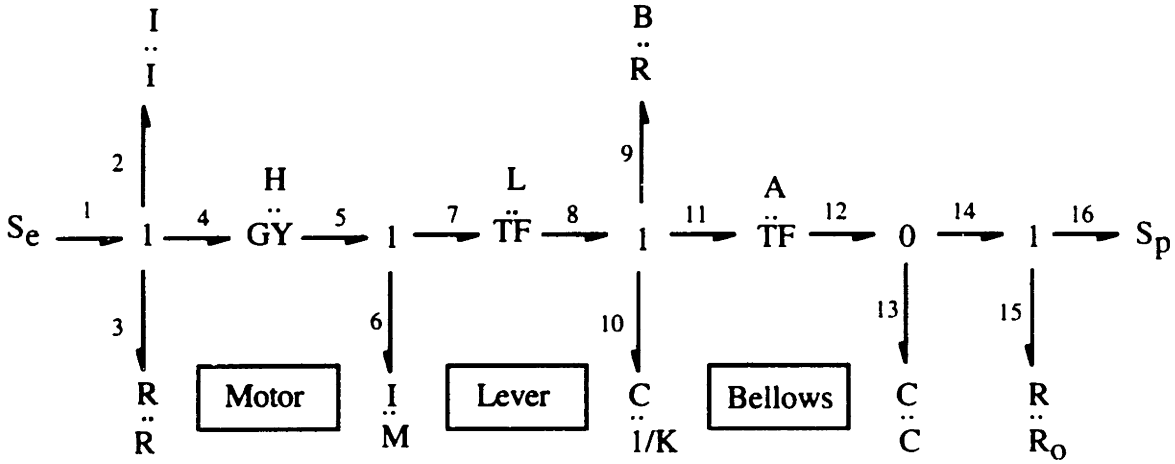


Figure 7.12 Bond graph representation of air pump

The dynamic characteristics of the above system can be obtained using commercial software like CAMAS™. Figure 7.13 shows the airflow at the outlet of the valve. This is the output of the simulation done in CAMAS™.

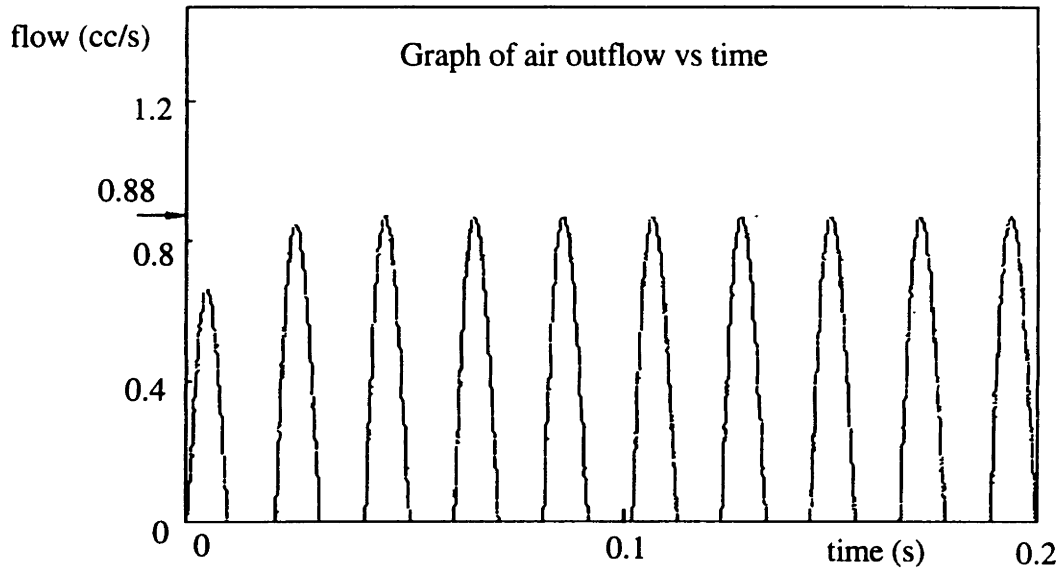


Figure 7.13 Air outflow from the air pump

7.3.2 Results

Different topologies generated by Variant are shown in Figure 7.14 and Figure 7.15. The pump system as represented by Figure 7.14 is simulated using the same set of parameter values; the output from the simulation is shown in Figure 7.16.

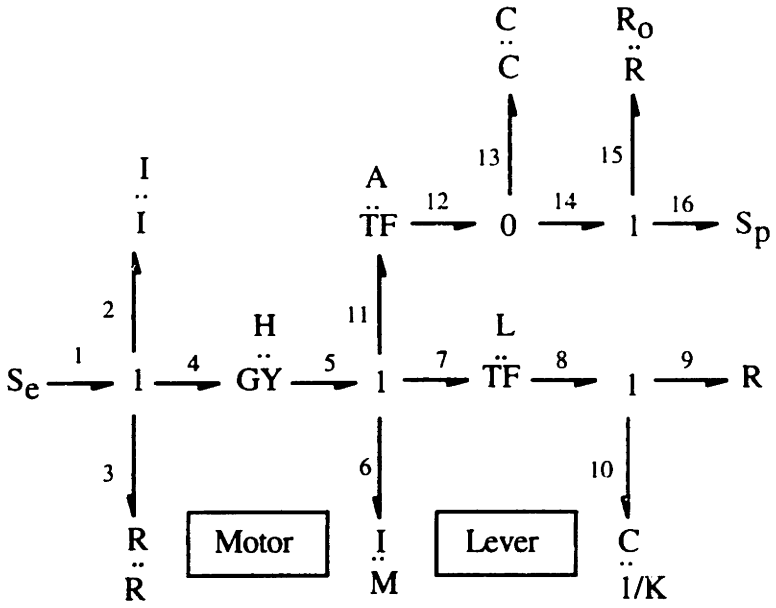


Figure 7.14 Modified bond graph from genetic algorithm

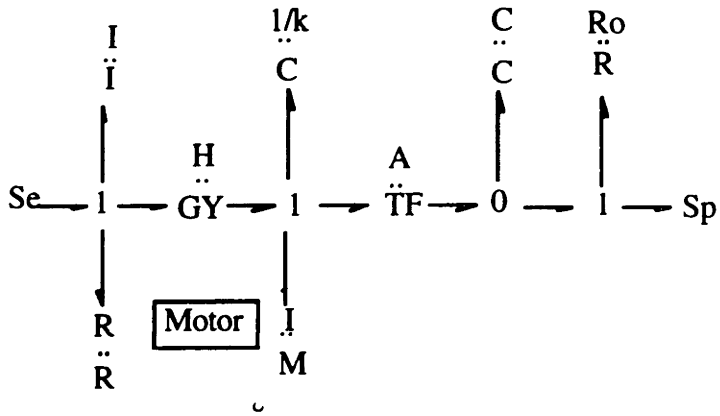


Figure 7.15 Modified graph from genetic algorithm

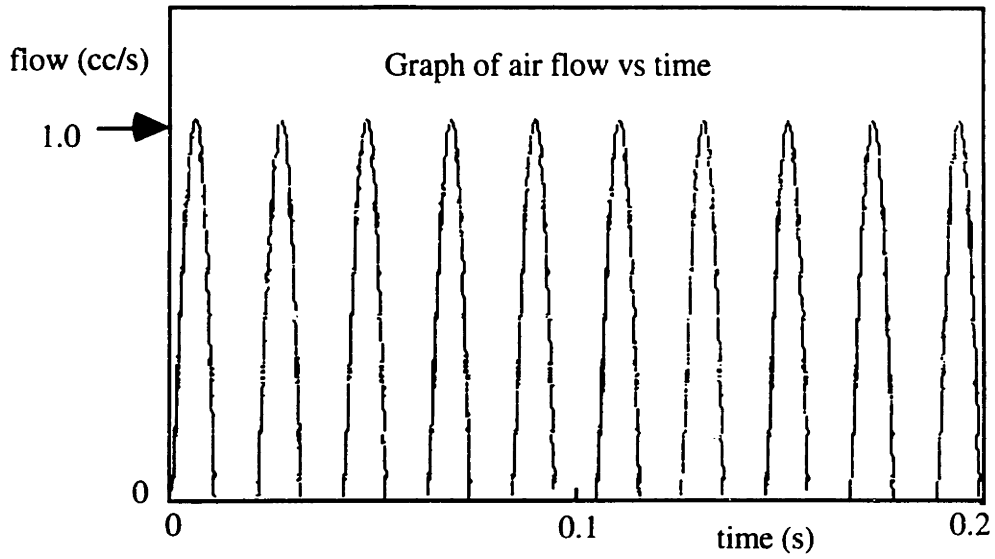


Figure 7.16 Air flow of the reconfigured bond graph

Thus, this modified pump is able to deliver a higher flow than the original pump. This is achieved through a reconfiguration of the pump elements. One of the possible physical realizations is shown in Figure 7.17.

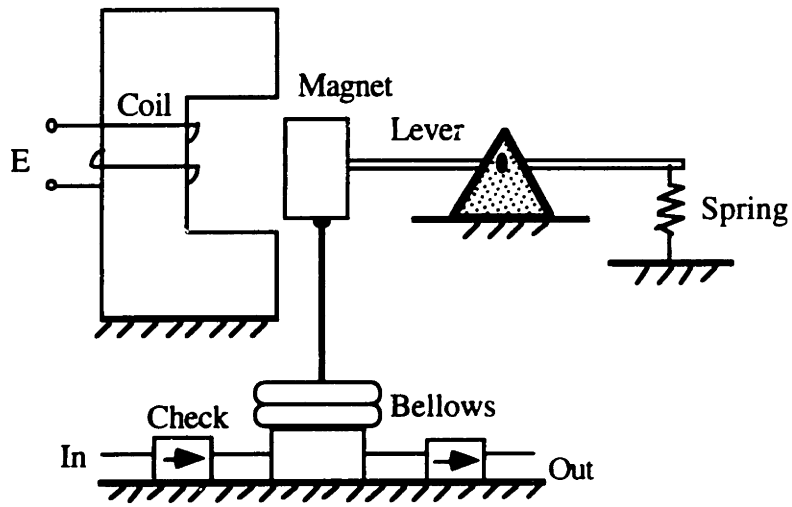


Figure 7.17 Physical realization of the bond graph

As for the other bond graph, more work needs to be done to translate it into schematic. One of the possible physical realizations is shown in Figure 7.18.

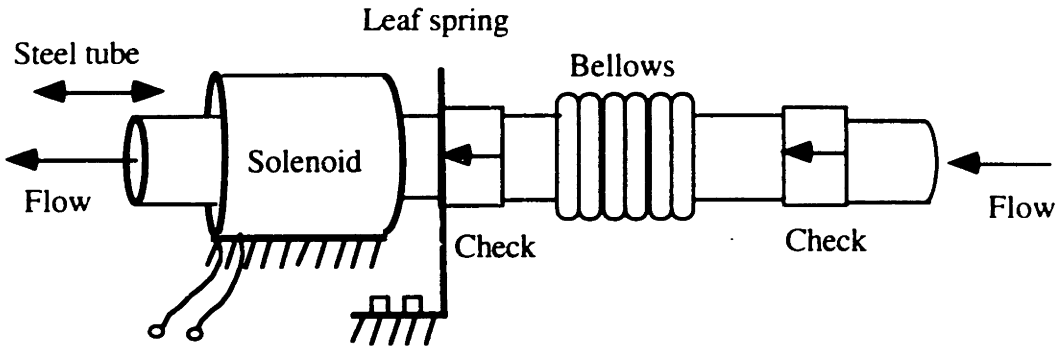


Figure 7.18 Design variant of pump

Chapter 8

Conclusions and Recommendations

8.0 Overview

This chapter describes the conclusions and the recommendations for future work. The conclusions are drawn from the discussions in the preceding chapters. The recommendations are made in preparation of applying this research to industry.

8.1 Conclusions

This thesis has explored the transformation in design that maps functional requirements to design descriptions. In particular, the author advocated the use of behavioral transformation and structural transformation to generate design variants. The model of innovative design-as-search in the problem-solution space has opened an avenue into the research of innovative designs.

Bond graphs have proved to be robust and versatile for modeling both problems and solutions. In this way, it is possible to traverse the problem-solution space, giving rise to new variants. Genetic algorithms have also been shown to be capable of providing the

operators needed in the evolution of design variants. Through various dynamic characteristics criteria, such as stability, phase and gain margins, a d transmissibility provided by MATLAB™, the computational platform has managed to generate a number of useful variants yielding the desired dynamic characteristics.

8.2 Recommendations for Future Work

The recommendation for future work includes many axes. In conceptual design, more work can be done on the synthesis of some of the design methods. This is especially true for the top-down design methods such as Suh's Axiomatic Method. Combining a top-down approach with an evolutionary method may be a good recipe for novel design generation.

On the bond graph axis, there is work to be done on inclusion of feedback information. Having said that, the writer must quickly add that the research will then take a turn toward controller design. This will add another dimension to design, as the computational platform will then be able to deal with power and signal flows.

On the genetic algorithm axis, research can be done on other strategies of speciation such as crowding, and monitoring the effects of convergence. Little sensitivity analysis has been done on the genetic parameters, as this was not the thrust of this research. However, now that feasibility has been demonstrated on the generation of variants, work can now be focused on the optimization of the genetic algorithm.

On the program level, more work can be done to conceal the bond graph interface. This may be useful for designers who are not familiar with bond graphs. Since bond graphs are abstractions, perhaps a more familiar user interface could be constructed which

shows the designer common elements of design, like resistors and capacitors in the electrical domain, instead of their bond graph equivalents.

References

[Agogino 87] Agogino, A.M. and A. S. Almgren, *Symbolic Computation in Computer-Aided Optimal Design* in Expert Systems in Computer-Aided Design, J.S. Gero, ed., 1987, Amsterdam, North-Holland, p. 267-284.

[Altshuller 84] Altshuller, G., *Creativity as an Exact Science*, 1984, New York, Gordon and Breach Science Publishers.

[Amsterdam 93] Amsterdam, J.A., Automated Qualitative Modeling of Dynamic Physical Systems, PhD Thesis, Dept. of Electrical Engineering and Computer Science, MIT, 1993.

[Anagnostou 92] Anagnostou, G., E. Ronquist and A. Patera, *A Computational Procedure for Part Design*, Computer Methods in Applied Mechanics and Engineering 92, 1992, p. 33-48.

[Balachandran 87] Balachandran, M. and J. S. Gero, A Knowledge-Based Approach to Mathematical Design Modeling and Optimization, Engineering Optimization, 1987, Vol. 12, p 91-115.

[Beaman 88] Beaman, J.J. and P.C. Breedveld, Physical modeling with Eulerian frames and bond graphs, Trans. ASME, J. Dynamic Syst. Measure. Control, Vol. 110(2), 1988, p. 182-188.

[Cagan 88] Cagan, J. and Agogino, A.M., *1stPrince: Innovative Design from First Principles*, 7th National Conference on Artificial Intelligence, AAAI-88, Minneapolis, MN, August 21-26, 1988.

[Chakrabarti 91] Chakrabarti, A. and T. Bligh, *Towards a decision-support framework for mechanical conceptual design.* in *ICED*, 1991, Zurich.

[Chapman 94] Chapman, C.D., *Structural Topology Optimization Via The Genetic Algorithms*, M.S. Thesis, Dept. of Mechanical Engineering, MIT, 1994.

[Chomsky 57] Chomsky, N., *Syntactic Structures*, Mouton, The Hague (Eight Printing, 1969).

[Cross 84] Cross, N., *Developments in Design Methodology*, Wiley, New York, 1984.

[Cutkosky 90] Cutkosky, M.R. and J.M. Tenenbaum, *A Methodology and Computational Framework for Concurrent Product and Process Design,* Mechanism and Machine Theory, 1990, **25**(3): p. 365-381.

[Dasgupta 94] Dasgupta, S., *Creativity in Invention and Design*, Cambridge University Press, U.K., 1994.

[

Davis 91] Davis, L., ed., *Handbook of Genetic Algorithms*, 1991, Van Nostrand Reinhold, New York.

[Deb 89] Deb, K. and D. Goldberg, *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, 1989, p. 42-50.

[Erner 93] Ermer, G., et al, *Steps Toward Integrating Function-Based Models and Bond Graphs for Conceptual Design in Engineering.* in *Automated Modeling for Design, ASME Winter Annual Meeting.* 1993.

[Finger 89] Finger, S. and J.R. Dixon, *A Review of Research in Mechanical Engineering Design. Part I: Descriptive, Prescriptive, and Computer-Based Models of Design Processes.* Research in Engineering Design, 1989, Vol. 1, p. 51-67.

[Finger 89] Finger, S. and J.R. Dixon, *A Review of Research in Mechanical Engineering Design. Part II: Representations, Analysis, and Design for the Life Cycle.* Research in Engineering Design, 1989, Vol. 1, p. 121-137.

[Finger 90] Finger, S. and J.R. Rinderle. *Transforming Behavioral and Physical Representations of Mechanical Designs.* in *Proceedings of the First International Workshop in Formal Methods in Design, Manufacturing and Assembly,* 1990, Colorado Springs.

[Gero 90] Gero, J.S., *Design prototypes: a knowledge representation schema for design,* in *A.I. Magazine,* 1990, p. 27-36.

[Gero 93] Gero, J. and B. Kumar, *Expanding design spaces through new design variables,* Design Studies, 1993, Vol. 14(3): p. 210-221.

[Goldberg 89] Goldberg, D., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Massachusetts.

[Gordon 61] Gordon, W.J.J., *Synectics: The Development of Creative Capacity*, 1st Ed., 1961, New York, Harper and Brothers.

[Grefenstette 86] Grefenstette, J., *Optimization of Control Parameters for Genetic Algorithms*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-16, 1986, p. 122-128.

[Harris 87] Harris, C.M., ed., *Shock and Vibration Handbook*, 3rd ed., McGraw-Hill, New York, 1987.

[Hauser 88] Hauser, J.R. and D. Clausing, *The House of Quality*, Harvard Business Review, 1988, (May-June).

[Hayes-Roth 85] Hayes-Roth, B., *A Blackboard Architecture for Control*, Artificial Intelligence, 1985, Vol. 26, p. 251-321.

[Holland 75] Holland, J.H., *Adaptation in Natural and Artificial Systems*, 1975, Ann Arbor, The University of Michigan Press.

[Hrovat 85] Hrovat, D. and W.E. Tobler, Bond graph modeling and computer simulation of automotive torque converters, J. Franklin Inst., Vol. 319(1/2), 1985, p. 93-114.

[Hrovat 82] Hrovat, D., A bond graph modeling of pneumatic passive and semi-active vehicle suspension, Int. J. Modeling Simulation, Vol. 2(1), 1982, p.16-21.

[Hundal 90] Hundal, M.S., *A Systematic Method for Developing Function Structures, Solutions and Concept Variants,* Mechanisms and Machine Theory, 1990, Vol. 25(3), p. 243-256.

[Karnopp 90] Karnopp, D.C., D.L. Margolis, and R.C. Rosenberg, *System Dynamics, A Unified Approach*, 2nd ed. 1990, John-Wiley and Sons, Inc.

[Kerley 87] Kerley, J. J., *Retroduction: A New Structured Approach to Mechanical Design.* in *ICED 87*, 1987, Boston, MA.

[Londono 89] Londono, F., K.J. Cleetus, and Y.V. Reddy, *A Blackboard Scheme for Cooperative Problem-Solving by Human Experts.* in *Computer-Aided Cooperative Product Development.* 1989, MIT, Cambridge, USA, Springer-Verlag.

[Maher 90] Maher, M.L., *Process Models for Design Synthesis,* in *A.I. Magazine,* 1990, p. 49-58.

[Mahfoud 95] Mahfoud, S.W., *Niching Methods for Genetic Algorithms*, PhD Thesis, Dept. of General Engineering, University of Illinois at Urbana-Champaign, 1995.

[Malmqvist 93] Malmqvist, J., *Computer-aided Conceptual Design of Dynamic Systems*, Technical Report No. 1993-06-30, Machine and Vehicle Design, Chalmers University of Technology, Goteborg, Sweden, 1993.

[Malmqvist 92] Malmqvist, J., *Computer-Aided Conceptual Design of Energy-transforming Technical Systems Based on Technical Systems Theory and Bond Graphs*, Technical Report No. 1992-12-29, Machine and Vehicle Design, Chalmers University of Technology, Goteborg, Sweden, 1992.

[Malmqvist 90] Malmqvist, J., *A Design System for Parametric Design of Complex Products*, in *Advances in Design Automation - 1990*, DE-Vol. 23(1), Book No. H0622A, ASME, 1992, p. 17-24.

[Margolis 79] Margolis, D.L. and D.C. Karnopp, Analysis and simulation of planar mechanism systems using bond graphs, ASME J. Mech. Des., Vol. 101(2), 1979, p. 187-191.

[Martens 72] Martens, H.R. and A.C. Bell, *A Logical Procedure for the Construction of Bond Graphs in Systems Modeling*, Journal of Dynamic Systems, Measurement, and Control, ASME, 1972, Vol. 94(3): p. 183-188.

[Martin 73] Martin, J., *Design of Man-Computer Dialogues*, 1973, Prentice-Hall, Englewood Cliffs.

[Mauldin 84] Mauldin, M.L., *Maintaining diversity in genetic search*, Proceedings of the National Conference on Artificial Intelligence, 1984, p. 247-250.

[Nelson 94] Nelson, F.C., *Vibration Isolation: A Review, I. Sinusoidal and Random Excitations*, Shock and Vibration, Vol. 1(5), 1994, p. 485-493.

[Newell 72] Newell, A. and H.A. Simon, *Human Problem Solving*. 1972, New Jersey, Prentice-Hall, Inc., Englewood Cliffs.

[O'Shaughnessy 93] O'Shaughnessy, K. and R.H. Sturges, *A Systematic Approach to Conceptual Engineering Design*. in *Proceedings of the 1993 NSF Design and Manufacturing Systems Conference*, 1993, Charlotte, North Carolina.

[Pahl 84] Pahl, G. and W. Beitz, *Engineering Design*, 1984, The Design Council.

[Papalambros 82] Papalambros, P.Y. and D.J. Wilde, *Principles of Optimal Design: Modeling and Computation*, 1982, Cambridge University Press, Cambridge, England.

[Paynter 61] Paynter, H.M., *Analysis and Design of Engineering Systems*, 1961, Cambridge, MA, MIT Press.

[Perkins 86] Perkins, D.N., *Knowledge as Design*. 1986, Lawrence Erlbaum Associates.

[Perkins 92] Perkins, D.N., *The Topography of Invention* in *Inventive Minds*, Weber, D.N., Ed., 1992, Oxford University Press.

[Peter 74] Peter, S.S., *Patterns in Nature*, 1974, Boston & Toronto, Little, Brown and Company.

[Prabhu 88] Prabhu, D.R. and D.L. Taylor, *Some Issues in the Generation of the Topology of Systems with Constant Power-Flow Input-Output Requirements.* in *Proceedings of the 1988 ASME Design Automation Conference*, 1988, Kissimmee, FL.

[Prabhu 89] Prabhu, D.R., *Synthesis of Systems from Specifications Containing Orientations and Positions Associated with Flow Variables.* in *Proceedings of the 1989 Design Automation Conference*, 1989, Montreal, Canada.

[Pugh 91] Pugh, S., *Total Design: Integrated Methods For Successful Product Engineering*, 1991, Addison-Wesley Publishing Company.

[Qian 93] Qian, L. and J.S. Gero, *Creative Engineering Design Using Analogy,* Computing in Civil and Building Engineering, ASCE, New York, 1993. Vol 2, p. 1634-1641.

[Ramaswamy 93] Ramaswamy, R., *Computer Tools for Preliminary Parametric Design,* 1993, MIT.

[Redfield 92] Redfield, R.C., *Bond Graphs As a Tool in Mechanical System Conceptual Design.* in *Automated Modeling*, 1992, CA, USA.

[Remmerswaal 85] Remmerswaal, J.A.M. and H.B. Pacejka, A bond graph computer model to simulate vacuum cleaner dynamics for design purposes, J. Franklin Inst., Vol. 319(1/2), 1985, p. 83-92.

[Rosenberg 71] Rosenberg, R.C., *State-Space Formulation for Bond Graph Models of Multiport Systems,* Trans. ASME Dynamic Systems, Measurement, and Control, 1971, Vol. 93(1): p. 35-40.

[Rosenberg 73] Rosenberg, R.C., *Modeling and Simulation of Large-Scale, Linear, Multiport Systems,* Automatica, 1973, Vol. 9, p. 87-95.

[Rosenberg 75] Rosenberg, R.C., *The Bond Graph as a Unified Data Base for Engineering System Design,* Journal of Engineering for Industry, 1975.

[Scrivener 93] Scrivener, S.A. et al, *Designing at a distance via real-time designer-to-designer interaction.* Design Studies, 1993, Vol. 14(3): p. 261-282.

[Serrano 87] Serrano, D., Constraint Management in Conceptual Design, PhD Thesis, Department of Mechanical Engineering, 1987, MIT.

[Slocum 92] Slocum, A.H., *Precision Machine Design*, 1992, Prentice-Hall,Inc.

[Sriram 89] Sriram, D. et al, *An Object-Oriented Framework for Collaborative Engineering Design.* in *Computer-Aided Cooperative Product Development*, 1989, MIT, Cambridge, USA, Springer-Verlag.

[Suh 90] Suh, N.S., *The Principles of Design,*. Oxford Series on Advanced Manufacturing, Crookall, M.C., ed., 1990, Oxford University Press.

[Suh 77] Suh, N.P., A.C. Bell and D.C. Gossard, *On an Axiomatic Approach to Manufacturing Systems, Journal of Engineering for Industry*, Vol. 100(2), May 1977.

[Tomiyama 89a] Tomiyama, T., *Meta-model: A key to Intelligent CAD Systems*, Research in Engineering Design, Vol 1(1), 1989, p. 19-34.

[Tomiyama 89b] Tomiyama, T., *Object Oriented Programming Paradigm for Intelligent CAD Systems*, Intelligent CAD Systems III, Eds., V.Akman, W. Hagen, P.J. Veerkemp, Springer Verlag, 1989, p. 3-16.

[Tomiyama 94] Tomiyama, T., T. Kiriyama and Y. Umeda, *Toward Knowledge Intensive Engineering* in *Proceedings of the 1994 Lancaster International Workshop on Engineering Design CACD '94*, 1994, Lancaster, U.K.

[Ulrich 87] Ulrich, K.T. and W.P. Seering, *Conceptual Design: Synthesis of Systems Components*, Intelligent and Integrated Manufacturing Analysis and Synthesis, ASME, New York, 1987, p. 57-66.

[Ulrich 88] Ulrich, K.T. and W.P. Seering, *Function Sharing in Mechanical Design*. in *7th National Conference on Artificial Intelligence, AAAI-88*, 1988, Minneapolis, MN.

[Ulrich 89] Ulrich, K. and W.P. Seering, *Synthesis of Schematic Descriptions in Mechanical Design*, Research in Engineering Design, 1989, Vol. 1(1), p. 5-18.

[Wallace 93] Wallace, D.R. and M.J. Jakiela, *Automated Product Concept Design: Unifying Aesthetics and Engineering,* IEEE Computer Graphics and Applications, 1993, Vol. 13(4), p. 66-75.

[Wallace 94] Wallace, D.R., *A Probabilistic Specification-based Design Model: applications to search and environmental computer-aided design*, PhD Thesis, Dept of Mechanical Engineering, MIT, 1994.

[Ward 87a] Ward, A. and W.P. Seering, *An Approach to Computational Aids for Mechanical Design*, Proceedings of the International Conference on Engineering Design, Boston, MA, Aug 17-20, 1987.

[Ward 87b] Ward, A. and W.P. Seering, *Representing Component Types for Design*, Proceedings of the 1987 Design Automation Conference, Boston, MA, Sep 27-30, 1987.

[Watters 88] Watters, B.G., R.B. Coleman, G.L. Duckworth, E.F. Berkman, A Perspective on Active Machine Mounts, Proceedings of the IEEE Conference on Decision and Control, 1988, Vol. 3, p. 2033-2038.

[West 92] West, H., Personal Communication, 1992.

[Westinghouse 84] Westinghouse, *Report on Life Cycle Costs,* 1984, Westinghouse Productivity and Quality Center, Pittsburgh, PA.

[Wiliiams 89] Williams, B.C., *Invention from First Principles via Topologies of Interaction,* PhD Thesis, 1989, MIT.

[Yasuhara 80] Yasuhara, M. and N.P.Suh, *A Quantitative Analysis of Design Based on Axiomatic Approach*, Computer Applications in Manufacturing Systems, ASME Winter Annual Meeting, 1980.

[Yoshikawa 87] Yoshikawa, H., General Design Theory as a Formal Theory of Design, IFIP WG 5.2 Workshop on Intelligent CAD Systems, October 1987.

[Yoshikawa 82] Yoshikawa, H., General Design Theory: Theory and Application, Proceedings of the Conference on CAD/CAM technology in Mechanical Engineering, Cambridge, MA, March 24-26, 1982.

[Zwicky 69] Zwicky, F., *Discovery, Invention, Research through the Morphological Approach.*, 1969, Toronto, Ontario, The Macmillan Company.