AN EVOLUTIONARY METHODOLOGY FOR CONCEPTUAL DESIGN

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SERKAN GÜROĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
MECHANICAL ENGINEERING

JULY 2005

Approval of the Graduate School of Natural and Applied Sciences

_____

Prof. Dr. Canan Özgen

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Doctor of Philosophy.

_____

Prof. Dr. Kemal İder

Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Doctor of Philosophy.

_____

Prof. Dr. Abdülkadir Erden

Supervisor

Examining Committee Members

Prof. Dr. Faruk Arınç	(METU, ME)	_____
Prof. Dr. Abdülkadir Erden	(METU, ME)	_____
Prof. Dr. Bülent E. Platin	(METU, ME)	_____
Assoc. Prof. Dr. Mehmet Önder Efe	(ETU, EEE)	_____
Asst. Prof. Dr. Zühal Erden	(ATILIM UNIV., IE)	_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Serkan Güroğlu

Signature :

ABSTRACT

AN EVOLUTIONARY METHODOLOGY FOR CONCEPTUAL DESIGN

GÜROĞLU, Serkan
Ph.D., Department of Mechanical Engineering
Supervisor: Prof. Dr. Abdülkadir ERDEN

July 2005, 174 pages

The main goal of this thesis is the development of a novel methodology to generate creative solutions at functional level for design tasks without binding solution spaces with designers' individual experiences and prejudices. For this purpose, an evolutionary methodology for the conceptual design of engineering products has been proposed.

This methodology performs evaluation, combination and modification of the existing solutions repetitively to generate new solution alternatives. Therefore, initially a representation scheme, which is generic enough to cover all alternatives in solution domain, has been defined. Following that, the evolutionary operations have been defined and two evaluation metrics have been proposed. Finally, the computer implementation of the developed theory has been performed. The test-runs of developed software resulted in creative alternatives for the design task. Consequently, the evolutionary design methodology presents a systematic design approach for less experienced or

inexperienced designers and establishes a base for experienced designers to conceive many other solution alternatives beyond their experiences.

Keywords: Conceptual design, design automation, evolutionary design, multi-objective optimization in design, complexity in design, creativity in design.

ÖZ

KAVRAMSAL TASARIMA EVRİMSEL BİR YAKLAŞIM

GÜROĞLU, Serkan
Doktora, Makina Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Abdülkadir ERDEN

Temmuz 2005, 174 pages

Bu tezin amacı, tasarımcının önyargı ve bireysel deneyimlerinden bağımsız olarak işlevsel düzeyde yaratıcı tasarım seçenekleri oluşturacak yeni bir yöntem geliştirmektir. Bu amaçla, mühendislik ürünlerinin kavramsal tasarımında kullanılmak üzere, evrimsel bir yöntem geliştirilmiştir.

Bu yöntem, yeni çözüm önerileri oluşturmak için mevcut çözümlerin değerlendirilmesini, birleştirilmesini ve uyarlanmasını sürekli olarak gerçekleştirir. Dolayısıyla, öncelikle çözüm kümesinde yer alan tüm seçenekleri ifade edebilecek bir gösterim şablonu tanımlanmıştır. Daha sonra, evrimsel işlemler tanımlanmış ve iki değerlendirme ölçütü önerilmiştir. Son olarak, geliştirilen kuramın bilgisayar ortamında uygulaması gerçekleştirilmiştir. Geliştirilen yazılım ile yapılan denemeler sonucunda, yaratıcı özellikler içeren tasarım seçeneklerinin üretildiği gözlenmiştir. Sonuç olarak, evrimsel tasarım yöntemi, az deneyimli ya da deneyimsiz tasarımcılar için sistematik bir tasarım

yaklaşımı, tecrübeli tasarımcılar için ise tecrübelerinin ötesinde çözüm seçenekleri oluşturabilecekleri bir ortam sunmaktadır.

Anahtar kelimeler: Kavramsal tasarım, tasarım otomasyonu, evrimsel tasarım, tasarımda çok-amaçlı optimizasyon kullanımı, tasarımda karmaşıklık, tasarımda yaratıcılık.

To my family

ACKNOWLEDGEMENTS

First of all, I am deeply indebted to my advisor, Prof. Dr. Abdülkadir Erden for his invaluable support, supervision, encouragement and insight throughout this study.

I would like to express my gratitude to Dr. Zühal Erden and Prof. Dr. Faruk Arınç for valuable discussions, their patience and precious suggestions. I would also like to thank to the other members of my dissertation committee, Assoc. Prof. Dr. Önder Efe and Prof. Dr. Bülent Platin for their technical counsel and critical reviews of this study.

I am particularly thankful for Onur Çetin, Ali Emre Turgut and Kutluk Bilge Arıkan for fundamental and theoretical discussions, cooperation and proof-readings. But above all these, I am grateful for their companionship and encouragement. Each generously devoted significant time and effort to my thesis, and their suggestions and comments led to substantial improvement in my thesis.

I would like to give many thanks Dilek Akbulut for fruitful discussions, her assistance and constant support.

Of course, none of this would have been possible without my family. I would like to thank them for their never ending understanding, help and support in every stage of my life.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER I

INTRODUCTION

1.1 BACKGROUND AND MOTIVATION

The engineering design activity is defined, in general, as a creative, iterative and often open-ended process of conceiving and developing components, systems and processes to satisfy real needs (Dixie, 2004). Due to the implicit and indirect nature of the relations between customer needs and design concepts, it is commonly accepted that this activity possesses artistic characteristics besides its scientific and methodic aspects. This sophisticated structure of design prevents the immediate formulation and standardization of it. In many design issues of engineering products, the design team members seek solutions directly basing on their knowledge and their past design practices. Therefore, the designer's biases and experiences have significant effects on the generated solutions.

Over the last twenty years, the researchers have focused on the studies aiming at placing design concept at a higher intellectual level and developing design as a discipline with its own structure, techniques and vocabulary. The design models developed by these studies can be classified into two main groups: the design process models and the design artifact models (Erden, 2004).

Design process models aim at revealing the methodology and characteristics of the engineering design process and proposing strategies for the designers about how to proceed in design procedure. One of the most important process

models proposed by Pahl and Beitz (1988) defines the flow of work at four main steps namely; the clarification of the task, conceptual design, embodiment design and detail design. The clarification of the task states the problem, objective and constraints, as well as the collection of information about the needs to be fulfilled. The second step, which is the conceptual design, includes transformation of the needs into functional requirements, creation of design alternatives and evaluation of them according to the design criteria. Next comes the embodiment design, which involves the transition from function structure of the design to form of product through the selection of appropriate components. Moreover, preparation of a preliminary layout, determination of production processes and optimization issues are performed in this step. Finally, detail design consists of all technical and economical calculations of the whole product and finalization of design. This flow of design process is presented in Figure 1.1.

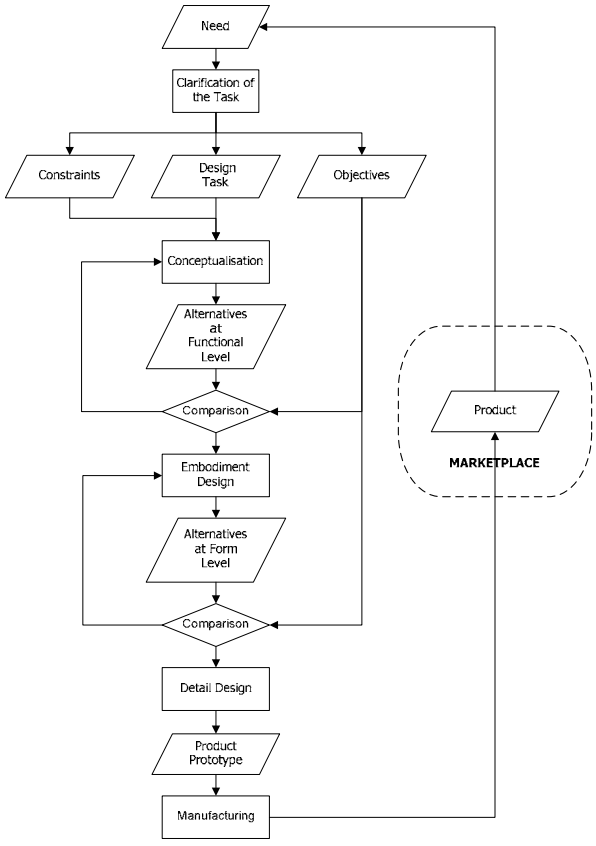


Figure 1.1. Flow of design process.

Among these steps, the conceptualization is the most vital and difficult part of design activity. During this stage, designers deal with the incomplete and informal information combining their intelligence and creativity to make large number of decisions and compare numerous alternatives. The studies that have been performed with a number of different industrial companies with complex products (e.g. advanced machine tools, motor vehicle), have indicated that the design of these products involve some 8-10 levels of decision making activity. Additionally, at each of these levels, the designers have to consider approximately 15 alternative function structures, embodiments or details. This geometric increase leads to about 2.5 billion decisions of purely qualitative nature, which necessitate a similar number of quantitative decisions or calculations (Sharpe, 1995).

Furthermore, these decisions made in early phases of conceptualization, have significant effects on whole design, manufacturing and even marketing activities. It is generally admitted that the first 5% of design process commits 80% of the overall cost to market the product (Carruba, 1993).

In the light of these facts, in order to model the products at conceptual and embodiment design stages, design artifact (or product) models have been developed. In design artifact models, researchers aim at extracting functional, structural and behavioral characteristics of the engineering systems regardless of their specific tasks (Erden, 2004). Expanding demands on the development of multi-technical artifacts (e.g. mechatronic products or systems) have increased the significance of such modeling approaches. The development of these products requires designers from different engineering disciplines to work together. Therefore, such a modeling methodology provides the designers with a powerful tool for correct, complete and systematic information transfer to the embodiment design, detail design and documentation phases (Erden, 1999).

Moreover, an efficient modeling technique with computer implementation highly reduces time and energy spent by the designers on construction and

evaluation of the design alternatives. The effect of modeling on the certainty with respect to the product performance is illustrated in Figure 1.2 (Soemers *et al.*, 2000). At the beginning of conceptualization, design freedom is maximum. However, certainty with respect to the performance and the costs is minimum. In the following phases of the design, while the design freedom decreases, certainty about the performance of the product increases. The decrease in the design freedom makes serial modifications on design very difficult. At this point, effective modeling increases the knowledge with respect to the performance as early as possible, and reveals the risks, opportunities and peculiarities of the design.

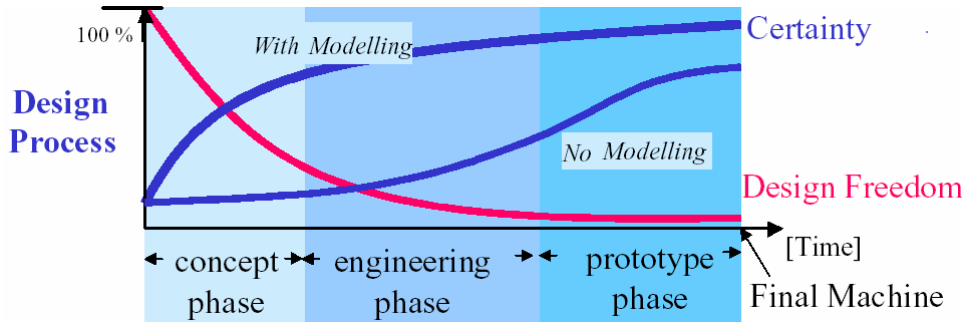


Figure 1.2.The effect of modeling on the certainty about the product performance (Soemers *et al.*, 2000).

The next step of these studies is the automation of design process. The modeling strategies mentioned above are limited to purposes of archival and transmittal of design information, or benchmarking on performances of developed alternatives. In addition to detection of design faults and suggestions for minor improvements, guiding designers in generation of novel designs or even performing the whole design process or some part of it without requiring the supervision of the designer are primary goals of ongoing research on design theory. Due to the integrated nature of the conceptualization phase including human intelligence and engineering creativity, automation of this phase is very difficult to achieve.

Although in today's engineering problems many design issues transform into concept evaluation (i.e. performance and cost assessment and optimization) rather than concept generation (Otto and Wood, 2001), creativity is always an essential characteristic of design activity and even designs are classified with respect to level of creativity involved (Bahrami and Dagli, 1994);

- Creative Design: Conceptualization of the problem begins with an abstract decomposition of the design task. If no priori decomposition exists for the solution of the problem, creative designs are taken into consideration.
- Innovative Design: If the design starts with a known decomposition, the design task will transform to the activity of uncovering appropriate component alternatives for the decomposition in hand.
- Adaptive design or Redesign: Modification of a completely known design to satisfy changes in the original function structure is defined as redesign.
- Routine Design: If both the priori decomposition of the design task and its solution alternatives are known in advance, the design issue becomes the determination of the most suitable alternative.

Thanks to the limited involvement of creativity in redesign and routine design activities, automation of these design types is accomplished to a certain extent through the use of various engineering analysis, CAD and CAE tools.

The recent researches on the design automation have mainly focused on innovative design and consequently numerous techniques and tools have been proposed. Rule based expert systems for electronic component design or software design, case based expert systems for mechanical or mechatronic design issues, genetic design tools for topology or configuration, evolutionary software design techniques, and data centric design databases / knowledge centric design repositories for system design are only some of them. Due to nature of innovative design based on appropriate component mining rather

than analysis and decomposition of design task, this design type becomes eligible for automation.

However, there are a small number of studies dedicated to the automation of creative design. Since creative designs require a complicated mental effort far beyond the combinatorial exploration of physical solution alternatives, building a methodology for the automation of creative design is much more difficult than that for the innovative design. For instance, in a design task, although preferring a chain-sprocket pair rather than a gear box generates a variance in solution, the functional composition behind both of these solutions are same and the main aim is to "increase torque" in a power transmission unit. Likewise, in many cases, the advances in both hardware and software technologies also create variety in designs but these new designs also become functionally equivalent of their initial prototypes (Roston, 1994). Since the functional resolution of the design task is the key point of the generation of the creative solutions (Ullman, 1997), a creative design methodology should be capable of generating different functional resolutions for the same design task.

The major motivation for this thesis is the fact that field of creative design is overlooked in design automation research in literature. Besides its complex nature, it is a probable consequence of the fact that many of today's engineering problems can be solved by the use of routine design, redesign and rarely innovative design activities. However, expanding demands on quality, increasing variety in customer needs, shrinking product cycles and budgets force the designers to generate creative solutions. Furthermore the intense competition conditions compel designers to respond changing trends in the market in a very short time. Therefore a new creative design methodology both for systemization and the automation purposes becomes a visible need.

1.2 THE SCOPE OF THE STUDY AND METHODOLOGY

This study aims at developing a design methodology to generate and identify creative solutions in early design process. Aside from the need for a creative design methodology in the solution of entirely new design problems, the proposed approach is a candidate for being an integrated part of the studies on innovative design automation. Since all innovative design issues require a priori decomposition of the design task, constitution of the original decompositions obtained through this methodology will help the generation of inventive products. Moreover, assurance of the creativity gives designers the opportunity of directing their efforts mostly on definition of new objective criteria.

In order to accomplish this goal, an abstract decomposition of the design task should be available at first. Therefore a functional modeling technique revealing working principles of artifact, basic functional modules and their interactions should be involved. Such a modeling technique not only translates the customer needs into a functional description, but also concentrates on how the product must carry out these functions to achieve the overall design task. For this purpose, in this study a functional modeling technique using *reconciled functional basis* proposed by Hirtz *et al.* (2001a) is employed.

Since the functional domain presents numerous solution possibilities, the next step is the development of a heuristic for the inference of functional structure or a search strategy for the exploration in functional domain. Although there exists some studies involving traditional search strategies (e.g. depth-first or breadth-first search etc.) (Malmqvist, 1995), in this study an evolutionary search strategy is employed. Unlike traditional search strategies, evolutionary techniques are not directionless. They utilize their past experiences to guide future events (Roston, 1994). Therefore, time consumption and possibility of reaching practically impossible solutions decreases. Employment of an evolutionary strategy also eliminates the need for the development of a functional reasoning technique. Evolutionary strategies do not need

understanding of procedures used to generate new individuals (i.e. new design alternatives in proposed methodology) (Roston, 1994). They only require a formal representation scheme for individuals and an objective function for the evaluation of them. Additionally, this feature makes the proposed methodology computer implementable.

Evaluation of generated designs at early phases is another serious problem met in construction of such a methodology. Due to abstract nature of conceptualization, it is very difficult to develop an evaluation metric that does not require human supervision. However automatic evaluation of the created design alternatives minimizes the dependency of designs to the designer's experience and the effects of designer's prejudices, intuitions and beliefs on the design decisions. Therefore in the scope of this study, two entropy based evaluation criteria to measure *the complexity* and *the creativity* of the generated designs are proposed.

Finally, the proposed methodology has been implemented in computer environment. The developed software has been tested through a case study on the design of household appliances.

1.3 OUTLINE OF THESIS

The remainder of the thesis is organized in six chapters. Chapter 2 presents an overview of engineering design approaches, review of design automation and modeling literature, exploration of the reasoning techniques and evolutionary search strategies employed in design automation and a survey on evaluation metrics for functional designs. Chapter 3 introduces the basics of evolutionary strategies. Chapter 4 gives the theoretical architecture of the developed evolutionary design methodology and proposed evaluation metrics. Chapter 5 provides computer implementation, case studies and results. Finally conclusions, discussions and possible future work are included in Chapter 6.

CHAPTER II

LITERATURE REVIEW

2.1 SURVEY ON METHODOLOGIES INVOLVED AT EARLY PHASES OF ENGINEERING DESIGN

The conceptual and embodiment design phases of product development period involve the completion of functional, structural (i.e. form of the product) and behavioral construction of the artifact (Edward *et al.*, 2000; Feng and Song, 2000). In order to systemize these significant design phases, two well-known problem solving strategies namely *bottom-up* and *top-down* design approaches have been suggested in the literature (Houstis, 2004). As the traditional approach, using bottom-up strategy produces solutions at physical level, top-down design strategy looks for original ideas at functional level before investigating physical solution alternatives. However, in order to provide designer with an appropriate level of abstraction and detail at each level, some combined strategies have also been developed.

In addition to the systematization of design process, these approaches also provide the designer with the ability of modeling designed artifacts at different levels of abstraction. These models are very beneficial, especially for taking immediate feedback on design decisions at different stages of the design process. This promises a comprehensive exploration of the design alternatives and a better realization of the design task (Sinha *et al.*, 2001).

Therefore many researchers have studied on the development, improvement and implementation of these approaches. The Sections 2.1.1, 2.1.2, 2.1.3 and 2.1.4 will provide a brief overview of these studies.

2.1.1 Top-Down Approach

Top-down design approach deals with *what we want to achieve*, rather than specifying physically *how we want to achieve it*. Therefore, in this approach, the design is driven from functional requirements toward solution alternatives (Terpenny, 1998). Some advantages of employing this approach can be summarized as follows (Otto and Wood, 2001);

- Derivation of functional requirements provides the designer a form independent expression of the design task. A comprehensive search for different form alternatives can be accomplished by the designer.
- In the cooperation of multi-disciplinary design team members, functional description provides a road map for the organization of tasks and processes. Meanwhile, interactions between independent functions clarify communication needed between concurrent design activities.
- Functional description of design task sets the boundaries to associate assemblies of subassemblies of final design solutions. Therefore resource allocation for concurrent engineering efforts is provided.
- Decomposition of the problems and seeking partial solutions improve creativity. Due to the reduction of extraneous information, solutions of sub-problems become more apparent.
- An important way of dealing with complexity is modularization (Rzevsky, 1995). Decomposition of the task provides modularization in design. It also helps the designer in fault detection and documentation.
- Abstraction leads the designer to solve the "real" problem rather than concentrating on particular solutions. Moreover, with the help of abstraction, it is possible to optimize the design according to various objectives such as reliability, energy consumption, ecological issues,

aesthetics, cost or manufacturability by considering trade offs through exchange of technologies (Rzevsky, 1995).

Before mentioning different methodologies developed for top-down approach, it should be considered that the distinctions between many of these are not so clear; they use some overlapping key concepts such as function, flow, constraint and objective.

In mathematics, a "function" is defined as a relation such that each element of a set is associated with a unique element of another set. It is obvious that in engineering design literature, perception of function concept differs. The selected definitions are as follows;

- Matousek (1963) views function as action required by the design problem (Chakrabarti, 1993).
- In Sembugamoorthy and Chandrasekaran's study (1986), a function is defined as the intended response of a device to external or internal stimuli (Chakrabarti, 1993).
- Pahl and Beitz (1988) state function as a general input-output relationship of a system whose purpose is to perform a task.
- Buur (1990) describes a function as the ability of a machine to create an expedient effect.
- Chakrabarti and Bligh (1992) identify function as a transformation between a set of (time-varying) input characteristics and a set of (time-varying) output characteristic.
- Ullman (1993) describes a function as transformation of objects and the relationships between them during an operational step. Moreover, this definition is expanded by covering the changes in relationships and object states that enable transformations.
- Function is defined by Blanchard and Fabrycky (1998) as the purposeful action performed by a system.
- According to the definition of Otto and Wood (2001), a function is a statement of a clear, reproducible relationship between the available

input and desired output of a product, independent of any particular form. Since its internal form is unknown, functions are represented by black boxes.

- In addition to these definitions, Suh (1990) has brought a new "Functional Requirement" expression into design theory. Functional requirements are defined as independent requirements that must be fulfilled in order to satisfy a certain need. This definition includes all functional and other life-cycle process properties. For instance, although it is not an ability of the system creating a physical effect, recycling is treated as a functional requirement. Moreover, parametric requirements are included by the functional requirement definition. This definition expands the boundaries of function definition.

These descriptions indicate that, although there is so far no generally agreed definition of the word "function", when these definitions are synthesized, the common properties of functions can be summarized as follows;

- A function is the ability of a system to create an external effect.
- A function should have an abstract formulation and it should be task independent.
- A function should not give a direction to the designer. As "resist bending" can be accepted as a function, "maintain toughness in bending" is not suitable due to its guidance to designer in solution generation (Salustri, 2000).
- In determination of a function, input-output relationships should also be considered.
- Each function should have at least one physically realizable solution.
- A function should be reproducible.
- In a function set, functions should be defined such that they are independent of each other. Redundant or overlapping function definitions should not be allowed.
- Life-cycle process properties may also be covered by this definition.

Although these properties enable the designer to identify customer needs entirely, this is achieved at the cost of a wider definition of the function, which causes difficulties in management of requirements. In order to overcome these difficulties, researchers have classified the functions into some sub-categories. Some of these classifications are presented in the following paragraphs.

Pahl and Beitz (1988) have classified functions as *main* and *auxiliary*. While the main functions serve the overall function directly, auxiliary functions contribute it indirectly. Auxiliary functions have a supportive or complementary character and are often determined by the nature of the solution. Some auxiliary functions are;

1. Communication functions involved to exchange of information between the system and the user or other systems,
2. Protection functions for protecting the main function against disturbances and protecting the environment against undesirable outputs,
3. Control functions to control main function and establish an interface between main and communication functions,
4. Power related functions to supply main function with required energy,
5. Structural functions to support components and modules and describe the spatial relationship between them.

Kusiak (2000) as cited by Korkmazel (2001) divides functions into 3 groups according to their related features;

1. Performance-related functions: Functions corresponding to the product performance requirements.
2. Process-related functions: Functions corresponding to the manufacturing process requirements.
3. Ergonomics-related functions: Functions corresponding to the ergonomic requirements.

Otto and Wood (2001) define 2 types of functions.

1. Basic functions: These are the overall product functions and represent the main reason behind the existence of the product; for example, the basic function of a CPU fan is to dissipate heat generated by CPU. Based on the conditions, a product may have more than one basic function. For example, the basic functions of an electric fuse are to "conduct electricity under certain conditions" and "break the circuit under certain other conditions".
2. Secondary functions: All other functions except basic ones are treated as secondary functions. There are three types of secondary functions; required, aesthetic and unwanted. One of the required functions of a cd player is to "generate rotary motion". The housing unit of it satisfies its aesthetic requirements. Additionally, a damping mechanism or rubber isolation may be employed to fulfill the requirement of "dampen vibration" unwanted by product.

Besides the categorization of functions regarding to their roles in the product, Chakrabarti (1997) has classified them with respect to their representation schemes in literature (Korkmazel, 2000).

- Verb-noun pair representation: The function is represented just a noun and an active verb such as "erase board", "transport item x". Although it is weak for computational support, it has informality and flexibility that designers need.
- I/O representation: Functions are described on the basis of inputs and outputs.
- Algorithmic representation: This representation is preferred especially in software design. Functions are arranged in a logical ordering. It is also suitable for the representation of decision type functions.
- State-based representation: In this representation, conversions between states are defined by functions.

Other concepts mentioned in developed methodologies are flows, constraints and objectives.

Flows are employed to define inputs and outputs of functions. Thus they reveal the relationships between functions or modules. Energy, matter and information are considered as basic concepts in any design problem (Stone and Wood, 2000). Conversions, variations or transmissions of these three concepts are followed by flows.

Constraints define the boundaries of allowable solution space (e.g. dimensions, maximum weight, etc.). Product must not violate these boundaries to be accepted as feasible (Malmqwist, 1995). Accepted solution principle determines the distinction on whether a customer need can be met with a function or should be considered as a constraint (Otto and Wood, 2001). As an example, if "compactness", which is a customer need, of a product is satisfied by a folding mechanism, then it becomes a function of the product. However, if the desired "compactness" is achieved by simply being small, here "compactness" becomes a constraint and a volumetric criterion that the entire product must satisfy (Otto and Wood, 2001).

Objectives form the basis for selecting the "best" solution among a number of design alternatives that fulfill all functional requirements and constraints (Malmqwist, 1995). Different methodologies propose different quality measures to evaluate the generated solution alternatives. According to the preferred methodology, objectives can change from "minimal cost", to "minimal information content". Some of these evaluation measures proposed for functional design are explained in Section 2.2.

All of these concepts are employed in the extraction and evaluation of functional model of designed artifact. These models can be constituted by either hierarchical function trees or graph based function structures.

2.1.1.1 Hierarchical Function Trees

The most effective way of deriving the hierarchical model of a product in sufficient depth is by modularizing the problem via a process known as "functional decomposition" (Ullman, 1997). Functional decomposition is defined as breaking an overall function into multiple sub-functions, which can be treated as individual sub-problems.

Two of the functional modeling techniques adopting hierarchical decomposition are;

- AND/OR Tree (Kusiak *et al.*, 1991a, 1991b)
- Functional Design Tree (Erden *et al.*, 1996)

Kusiak *et al.* (1991a; 1991b) have used AND/OR tree to represent design specifications. In this representation, design requirements are divided into sub-requirements up to reaching a corresponding function. In the functional space, each function specified for a given requirement can be further decomposed into sub-functions. This decomposition continues until reaching a possible match of the component. The relationships among requirements and functions at the same abstraction level are revealed using AND/OR links. AND/OR tree for a hypothetical product is given in Figure 2.1. AND links are represented by connecting arcs between same level requirements and functions, whereas no arcs stands for OR links.

This methodology has been primarily developed for the "requirement analysis", therefore objectives and constraints in the product design are not considered in the construction of the AND/OR tree. Moreover, no reasoning technique has been proposed for the extraction of requirements, functions and relations (mapping) between them. An evaluation methodology is another deficiency of this approach.

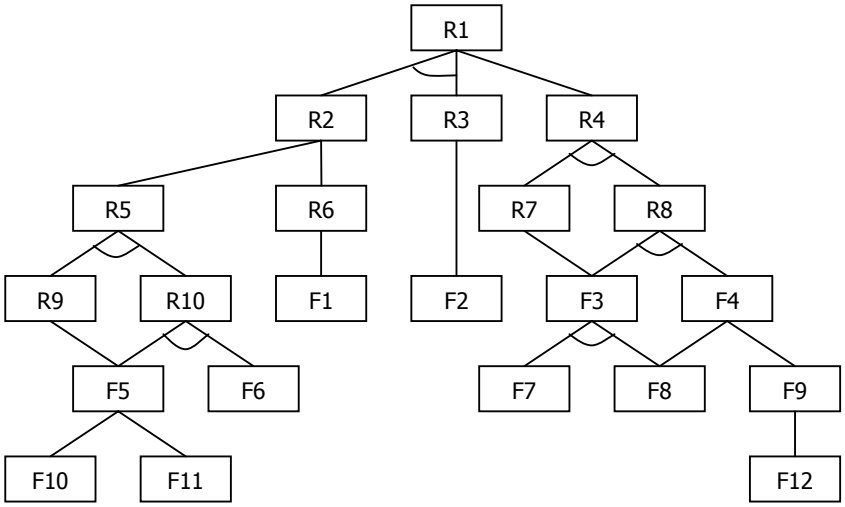


Figure 2.1 AND/OR tree for a hypothetical product (Kusiak *et al.*, 1991a).

Another hierarchical representation named "Functional Design Tree" has been mentioned in Erden *et al.*'s work (1996). It is a functional decomposition hierarchy that involves sub-functions of systems at various levels of resolution and where the top most node is to satisfy the required overall function (Erden *et al.*, 1999). In this representation, as one proceeds to the lower levels of tree, nodes gain precision in their definition. Nodes in the last level of decomposition can be defined numerically or in a formula-driven formal way representing precise sub-functions as precise input/output mappings.

A similar representation method has been employed in Otto and Wood's studies (2001). As recommended in these studies, this type of decomposition is very useful especially in reverse engineering activities. Generated function tree for a coffee mill is illustrated in Figure 2.2.

Although functional resolution of an artifact can be extracted by using subtract and operate procedure in reverse engineering activities, this methodology does not offer any reasoning technique for functional decomposition in design of novel products. Disregarding objectives and constraints and need of an evaluation methodology are other problems of this approach to handle.

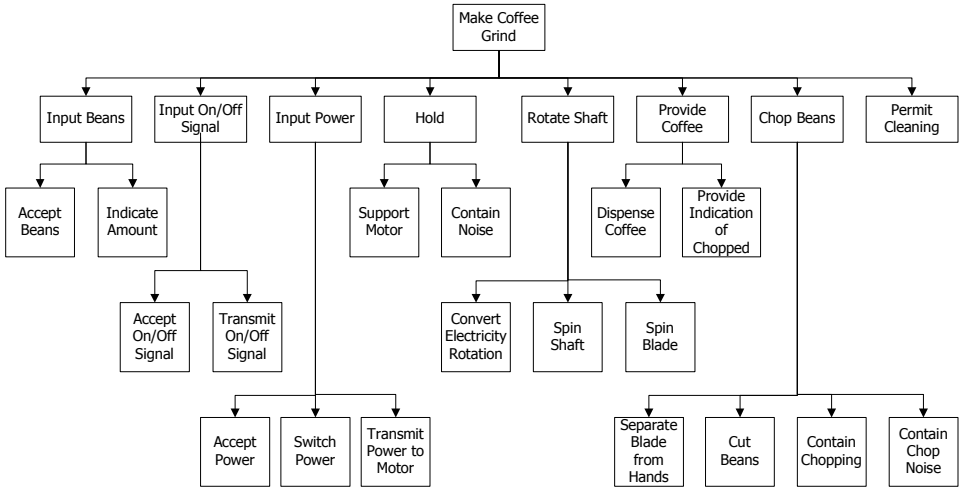


Figure 2.2 Function tree of coffee mill (Otto and Wood, 2001).

2.1.1.2 Graph Based Function Structures

Hierarchical function trees are very important representation schemes both for describing architecture of a product and determining level of abstraction that the designer handles the design task. However there is still a serious need for revealing the relationships between defined functions and modules in a product. For this purpose, a graph-based representation scheme has been proposed in the literature. The approaches using graph-based representation can be classified into three with respect to their need for a standard function/flow language. First approach does not use a standard language, the second approach requires a limited vocabulary and the third approach works with comprehensive standard languages.

One of the important studies in the first approach employs Functional Analysis Systems Technique (FAST) Diagrams in the representation of designed artifacts (Miles, 1972). Although these studies are performed mainly for value engineering purposes, in order to estimate the value of the product, they need to break the overall design into sub assemblies. The FAST Diagram of an overhead transparency projector is given in Figure 2.3.

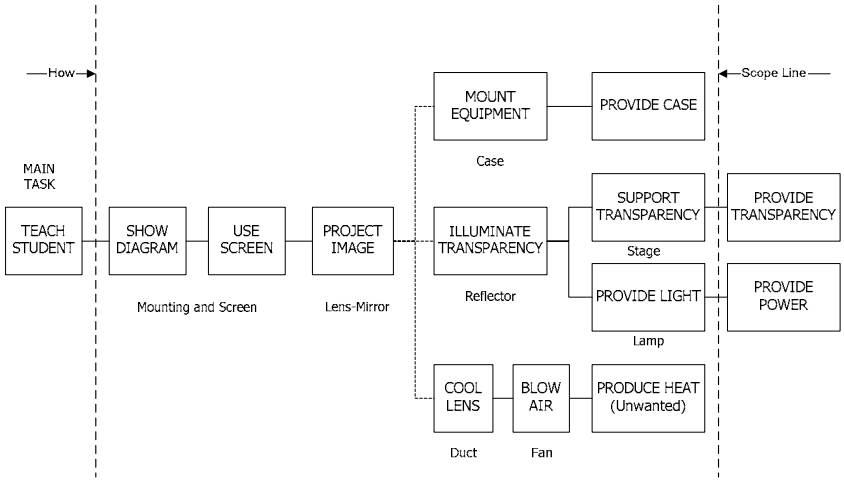


Figure 2.3 FAST Diagram of overhead transparency projector (Miles, 1972).

Some of the studies in the second approach aim at extracting function definitions of some specific problems and therefore, their functional languages have become limited to the scope of those specific problems (e.g. Collins *et al.* (1976) developed a vocabulary to accurately communicate helicopter failure information). However, Systematic Design Approach of Pahl and Beitz (1988) is different from these studies and it is applicable to general design problems. Although it has a limited vocabulary for defining functions and flows, Systematic Design Approach provides designers with a standard way to reveal sequential and parallel function chains and external/internal connections of these chains in a product.

In the third approach, there are two major attempts to generate a common design language namely NIST Design Repository project (Szykman *et al.*, 1999a; 1999b; 2000a; 2000b; Allen *et al.*, 2000) and Functional Basis (Little *et al.*, 1997; Otto and Wood, 1997; Stone and Wood, 1999; Stone and Wood, 2000; Kurfman *et al.*, 2001). The third research is on reconciliation of these two studies; Reconciled Functional Basis (Hirtz *et al.*, 2001a; 2001b).

In order to generate repeatable and meaningful designs, a standard set of functions and flows is essential. In particular, the development of a universal language, which defines product functions, will reduce the ambiguity at

modeling level. As the source of ambiguity can be a function/flow, which has multiple meanings accepted by different engineers or it can be a group of different functions/flows used to mean same entity by different designers.

Since the functions/flows are atoms of the modules, standard definition of them provides designers with a base for transition from integral architecture to modular architecture in product development. Meanwhile, product comparison and benchmarking activities can also be possible by development of the evaluation metrics for standard function structures. Overlapping function/flow definitions are the main reason of redundant designs. Such a standardization prevents designers from inferring redundant function structures.

For these purposes, a research has been driven by industry needs that were identified at a workshop held at NIST in November, 1996 (Szykman, 1999b). The aim of this study is to develop a repository to provide the effective reuse of design knowledge. In order to prevent a combinatorial explosion in function definitions, functions and flows are identified by generating two separate taxonomies. These taxonomies have broken down to reach atomic definitions. Thus, over 100 flows and 130 functions are identified while preserving the taxonomy generic enough due to the hierarchical decomposition. In addition, a concurrent effort has also been performed to constitute a language for obtaining functional blueprint of designed products by two US universities and their industrial partners. At the end of this study, a Functional Basis has been developed and applied to product similarity computations, functional tolerancing and design by analogy studies (Hirtz *et al.*, 2001b).

There is a high degree of similarity between these two studies. Therefore a reconciliation process has been performed to obtain a complete set of function and flows. Besides the same terms in the function and flow lists, there exist some synonyms and some others terms, which are specific to its basis, in these two research efforts. By considering these, a new set comprising mutually exclusive terms has been acquired (i.e. Reconciled Functional Basis).

This new basis includes a three level hierarchy; class (primary), secondary and tertiary. Specification of the level increases with the level number. Thanks to this hierarchy, the reconciled functional basis is generic enough to cover engineering design activities in many scales. Although it focuses primarily on the mechanical and electromechanical domains, the basis allows updating and adding new descriptors without disturbing the developed taxonomy. Therefore this makes the basis applicable to all disciplines.

Although reconciled functional basis supports other representation formats, verb-object representation is adopted in many studies. In this representation, verb part indicates the function and object part specifies the flow descriptor. Both of the function and the flow can be selected from any of the three levels depending on the specification desired. These function and the flow sets are presented in Tables 2.1 and 2.2 respectively.

In practice, a secondary flow is described by a secondary descriptor and a class descriptor such as biological energy. Tertiary flows are described by a tertiary descriptor and a class descriptor (e.g. solar energy). "Correspondents" category is not the fourth level of reconciled functional basis. The terms placed in this column are means of the terms described in reconciled functional basis. Secondary and tertiary flows may be further specified by adding a power conjugate complement. Here the flow description is formed by a secondary descriptor (or tertiary descriptor) and a complement. For instance, human energy description can be specified further with the description of human force. Detailed descriptions of the terms in function and flow lists are given in Appendix A.

In the extraction of function structures, it should be considered that the reconciled functional basis consists of *device functions* rather than *user functions*. For instance, a coffee maker imports the flow of water while a person pours water into the coffee maker.

Table 2.1 Functional basis reconciled function set (Stone *et al.*, 2001).

Class Primary	**Secondary**	**Tertiary**	**Correspondents**
Branch	Separate		Isolate, sever, disjoin
		Divide	Detach, *isolate*, release, sort, split, disconnect, subtract
		Extract	Refine, filter, purify, percolate, strain, *clear*
		Remove	Cut, drill, lathe, polish, sand
	Distribute		Diffuse, dispel, disperse, dissipate, diverge, scatter
Channel	Import		Form entrance, *allow*, input, *capture*
	Export		Dispose, eject, *emit*, empty, *remove*, destroy, eliminate
	Transfer		Carry, deliver
		Transport	Advance, lift, move
		Transmit	Conduct, convey
	Guide		Direct, shift, steer, straighten, switch
		Translate	Move, relocate
		Rotate	Spin, turn
		Allow DOF	*Constraint*, unfasten, unlock
Connect	Couple		Associate, connect
		Join	Assemble, fasten
		Link	Attach
	Mix		Add, blend, coalesce, combine, pack
Control Magnitude	Actuate		Enable, initiate, start, turn on
	Regulate		
		Increase	*Allow*, open
		Decrease	Close, delay, interrupt
	Change		Adjust, modulate, *clear*, demodulate, invert, normalize, rectify, reset, scale, vary, modify
		Increment	Amplify, enhance, magnify, multiply
		Decrement	Attenuate, dampen, reduce
		Shape	Compact, compress, crush, pierce, deform, form
		Condition	Prepare, adapt, treat
	Stop		End, halt, pause, interrupt, restrain
		Prevent	Disable, turn off
		Inhibit	Shield, insulate, protect, resist
Convert	Convert		Condense, create, decode, differentiate, digitize, encode, evaporate, generate, integrate, liquefy, *process*, solidify, transform
Provision	Store		Accumulate
		Contain	*Capture*, enclose
		Collect	Absorb, consume, fill, reserve
	Supply		Provide, replenish, retrieve
Signal	Sense		Feel, determine
		Detect	Discern, perceive, recognize
		Measure	Identify, *locate*
	Indicate		Announce, show, denote, record, register
		Track	Mark, time
		Display	*Emit*, expose, select
	Process		Compare, calculate, check
Support	Stabilize		Steady
	Secure		*Constrain*, hold, place, fix
	Position		Align, *locate*, orient

Table 2.2 Functional basis reconciled flow set (Stone *et al.*, 2001).

Class Primary	**Secondary**	**Tertiary**	**Correspondents**	
Material	Human		Hand, foot, head	
	Gas		Homogeneous	
	Liquid		Incompressible, compressible, homogeneous	
	Solid	Object	Rigid-body, elastic-body, widget	
		Particulate		
		Composite		
	Plasma			
	Mixture	Gas-gas		
		Liquid-liquid		
		Solid-solid	Aggregate	
		Solid-Liquid		
		Liquid-Gas		
		Solid-Gas		
		Solid-Liquid-Gas		
		Colloidal	Aerosol	
Signal	Status	Auditory	Tone, word	
		Olfactory		
		Tactile	Temperature, pressure, roughness	
		Taste		
		Visual	Position, displacement	
	Control	Analog	Oscillatory	
		Discrete	Binary	
Energy			**Power Conjugate Complements**	
			Effort Analogy	Flow Analogy
	Human		Effort	Flow
	Acoustic		Force	Velocity
	Biological		Pressure	Particle velocity
	Chemical		Pressure	Volumetric flow
	Electrical		Affinity	Reaction rate
	Electromagnetic	Optical	Electro-motive force	Current
		Solar	Effort	Flow
	Hydraulic		Intensity	Velocity
	Magnetic		Intensity	Velocity
	Mechanical	Rotational	Pressure	Volumetric flow
		Translational	Magneto-motive force	Magnetic flux rate
	Pneumatic		Effort	Flow
	Radioactive/ Nuclear		Torque	Angular velocity
	Thermal		Force	Linear velocity

As an example, function structure of an electric wok developed using reconciled functional basis is illustrated in Figure 2.4.

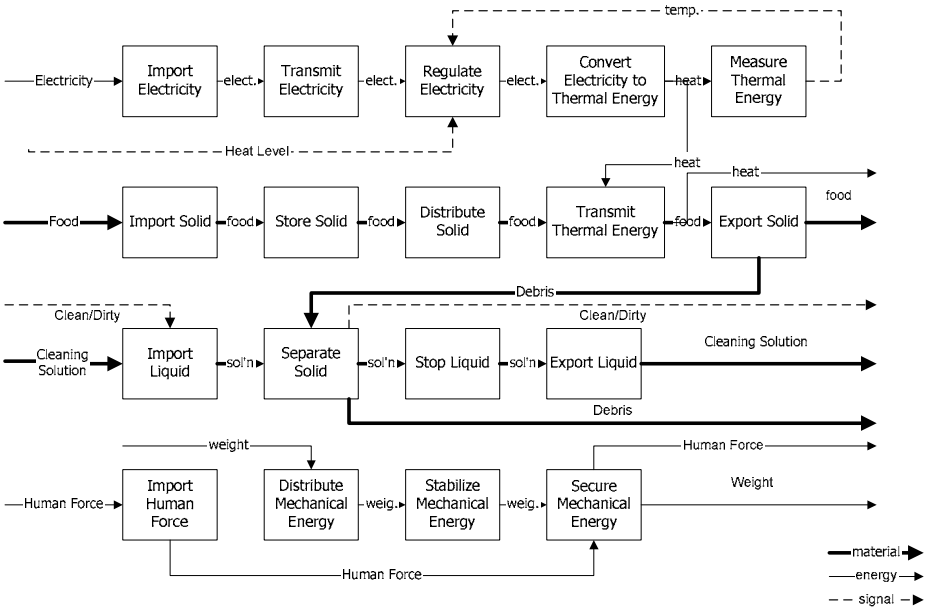


Figure 2.4 Function structure of an electric wok (Stone *et al.*, 2004).

It is obvious that there are many other attempts to systemize and model the top-down design approach. Only some selected researches in the development of functional design theory are mentioned here. A summary of these efforts and some others are presented in Table 2.3.

2.1.2 Bottom-Up Approach

Traditional engineering design tasks are handled through bottom-up approach. This approach drives designs from form to function. Satisfaction of functional requirements is an evaluation measure rather than a starting point. Bottom-up approach carries out the design activities at component level. In employment of this approach, as the physical realization of designed system is guaranteed, there is no immediate assurance that functional requirements are met on the first attempt unless the system is simple. Therefore this approach is known to be highly iterative (Terpenny, J. P., 1998).

Table 2.3 Overview of some important efforts in functional design.

Study/Reference	Goal	Representation	Challenges
Value Engineering (Miles, 1972; Akiyama, 1991; VAI, 1993)	Minimization of cost	Verb-Object FAST Diagram	No standard list for verbs and objects (Stone and Wood, 1999).
Failure Experience Matrix (Collins *et al.*, 1976)	Communicate helicopter failure information.	105 unique mechanical functions	Limited to helicopter systems (Stone and Wood, 1999).
TIPS/TRIZ (Altshuller, 1984)	Inventive Problem Solving, Value Engineering Analysis	30 functional descriptions	Does not cover whole design process (only inventive part is considered) (Malmqvist *et al.*, 1996).
Systematic Design Approach (Pahl and Beitz, 1988)	Modeling and Systematization	5 generally valid functions and 3 flows	High level of abstraction, Limited function/flow vocabulary.
A Systematic Method for Function Structures (Hundal, 1990)	Modeling and Systematization	6 functional classes and more specific functions for each class	For mechanical design applications, Limited function/flow vocabulary (Stone and Wood, 1999).
Axiomatic Design (Suh, 1990)	Modeling and Systematization	FR/DP Trees	No standard list for functions and flows
A Novel Approach for Decomposition (Kusiak *et al.*, 1991)	Modeling and Systematization	AND/OR Trees	No standard list for functions and flows.
Design Methodology (Grabowski and Benz, 1991)	Modeling and Systematization	Frame Based Function Description	For mechanical design applications (Erden, 1999).
Mechanical Design Process (Ullman, 1992)	Modeling and Systematization	Description of functions in terms of flows	For mechanical design applications (Erden, 1999).
Design Methodology (Koch *et al.*, 1994)	Systematization	20 sub-system representations	For mechanical design applications (Stone and Wood, 1999).
Classification of Functions (Kirschman and Fadel, 1998)	Systematization	4 basic mechanical function groups	For mechanical design applications (Stone and Wood, 1999).
The NIST Design Repository Project (Szykman *et al.*, 1999)	Creation of a design repository	130 functions and 100 flows	Some deficiencies in function-flow list.
Functional Basis (Little *et al.*, 1997; Otto, Wood, 1997)	Modeling and Systematization	8 function and 3 flow classes	Some deficiencies in function-flow list.
Reconciled Functional Basis (Hirtz *et al.*, 2001a)	Reconciling of NIST Design Repository and Functional Basis	8 function and 3 flow classes	Requirement for a formal computable form (Stone and Wood, 2000).

Bottom-up design is simply the physical synthesis of components. Morphological analysis is one of the key methodologies in concept generation through bottom-up approach (Weber and Condoor, 1998). Morphological analysis is first proposed by Zwicky (1948) as a method for identifying and investigating the total set of possible relationships or configurations contained in a given problem (Ritchey, 2002). All these relationships and configurations are defined at the level of shape or form.

In order to construct a morphological matrix, attributes of the product sought by designer are listed at first. These attributes can be parts, properties such as material, color etc. or design elements. They determine the columns of morphological matrix. Each row in the matrix is filled with possible variations of these attributes. Then the designer selects a combination of these variations to complete the design process.

Morphological matrix for a lamp design is constructed in Table 2.4. According to this matrix, some possible solutions can be

- A flashlight design with attributes of battery powered, low intensity and hand held or,
- A mains powered, medium intensity, daylight bulb, which can be used especially in clothes shops to allow customers to see the true color of clothes.

Table 2.4 Lamp design via morphological matrix (MindTools, 2004).

Power Supply	**Bulb Type**	**Light Intensity**	**Size**	**Style**	**Finish**	**Material**
Battery	Halogen	Low	Very Large	Modern	Black	Metal
Mains	Bulb	Medium	Large	Antique	White	Ceramic
Solar	Daylight	High	Medium	Roman	Metallic	Concrete
Generator	Colored	Variable	Small	Art Nouveau	Terracotta	Bone
Crank			Hand held	Industrial	Enamel	Glass
Gas				Ethnic	Natural	Wood
Oil/Petrol					Fabric	Stone
Flame						Plastic

Snavely *et al.* (1990) cited two other bottom-up design attempts in his study. The first one namely EDISON intends to obtain innovative designs by applying heuristics to generate new structural topologies (Dyer and Flowers, 1984; Dyer *et al.*, 1986). This study is one of the early works concentrating on cognitive area to generate creative designs. Such studies aim at mimicking human way of thinking, when designing and require an expert system. Although EDISON has generated some novel solutions for design tasks such as can opener or door design, application of it is limited to these highly simplified designs (Bently and Wakefield, 1996).

The second approach is Ward and Seering's mechanical design compiler (1989a; 1989b). Besides developing a heuristics, construction of a component database is another common method employed by bottom-up design approach. Mechanical design compiler uses a catalog of artifacts (i.e. off-the-shelve components) to satisfy a user defined topology. This compiler simply imports a schematic of a mechanical or hydraulic power transmission system and returns catalog numbers from predefined catalogs for the optimal selection of components implementing the design.

Due to their form based nature of adaptive design-redesign or routine design activities, bottom-up approach becomes more applicable. While creative design activities have the potential to generate infinite number of topologies, routine designs deal with a fixed topology (Snavely *et al.*, 1990). Although it is possible to change functional topology by changing physical means, in order to obtain creative solutions, it looks better starting from topology construction (i.e. top-down approach).

2.1.3 Combined Approach

These two approaches mentioned above indicate that as each designed functional topology can have more than one physical solution, design tasks can also be satisfied more than one functional topology. In order to reveal the relation between the functional decomposition and physical manifestations of

them, Hubka (1976) and Andreasen (1980) formulate the law of vertical causality. This law states that, hierarchical decomposition of a function is only possible, when a means (e.g. technical principle, an organ or a component etc.) has been chosen to realize that function (Buur, 1990). In this way, once a function is formulated, a number of alternative means to perform this function should be defined. Regarding to selected means a different set of subfunctions on a lower level will be involved. Therefore as functional resolution is completed, physical realization of artifact is also constructed through this combined approach. Two studies using this approach are mentioned in Section 2.1.3.1 and 2.1.3.2.

2.1.3.1 Axiomatic Design Theory

- Suh (1990; 1998; 2001), the developer of the axiomatic design theory, divides the design world into four domains illustrated in Figure 2.5. As the domains on the left represent goals, the domains on the right represent the design solutions to achieve them. Suh states that design is simply generation of mapping between these domains.

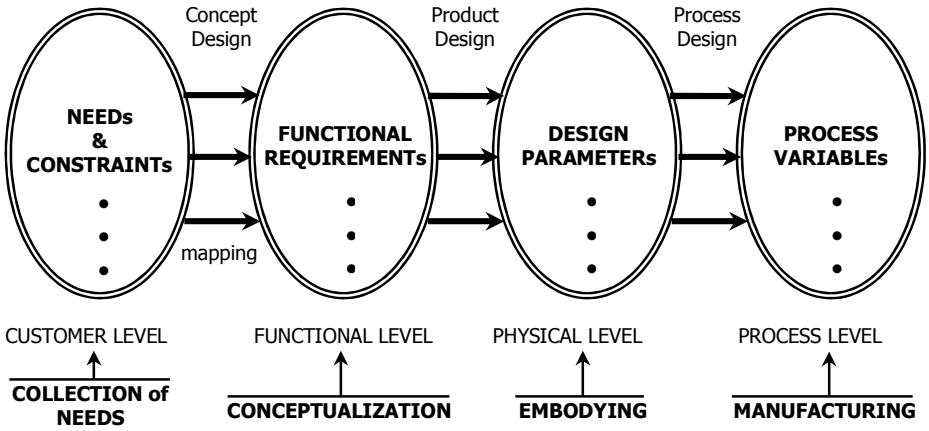


Figure 2.5 Four domains of design world.

This theory is based on two fundamental axioms that govern this mapping process.

- The Independence Axiom: Maintain the independence of functional requirements.
- The Information Axiom: Minimize the information content.

Axiomatic design theory classifies designs as uncoupled, decoupled and coupled with respect to dependencies of their functional requirements. As the first axiom guides the designer to generate uncoupled designs, the second axiom evaluates the generated designs regarding their information content. Information content is measured by an entropy based evaluation metric and it indicates the probability of success of a design. This metric needs the identification of design parameters at physical level. Identification process is performed by a zigzagging manner together with the decomposition of functional requirements. In addition to these axioms, some other corollaries and theorems are proposed in axiomatic design theory.

2.1.3.2 Function/Means Tree

Function/Means tree is basically a breadth-first search of a design space containing all possible solutions to a problem. Breadth-first searching is employed to help controlling the size of the design space (making the design easier to manage).

By operating the rules of vertical causality, function/means hierarchy is constructed layer by layer. Figure 2.6 illustrates a function-means diagram for a cigarette lighter. In this diagram, functions are represented by boxes. The ovals represent some possible means by which the top-level function can be achieved. For demonstration purpose, only two of the possible means are worked out in further detail.

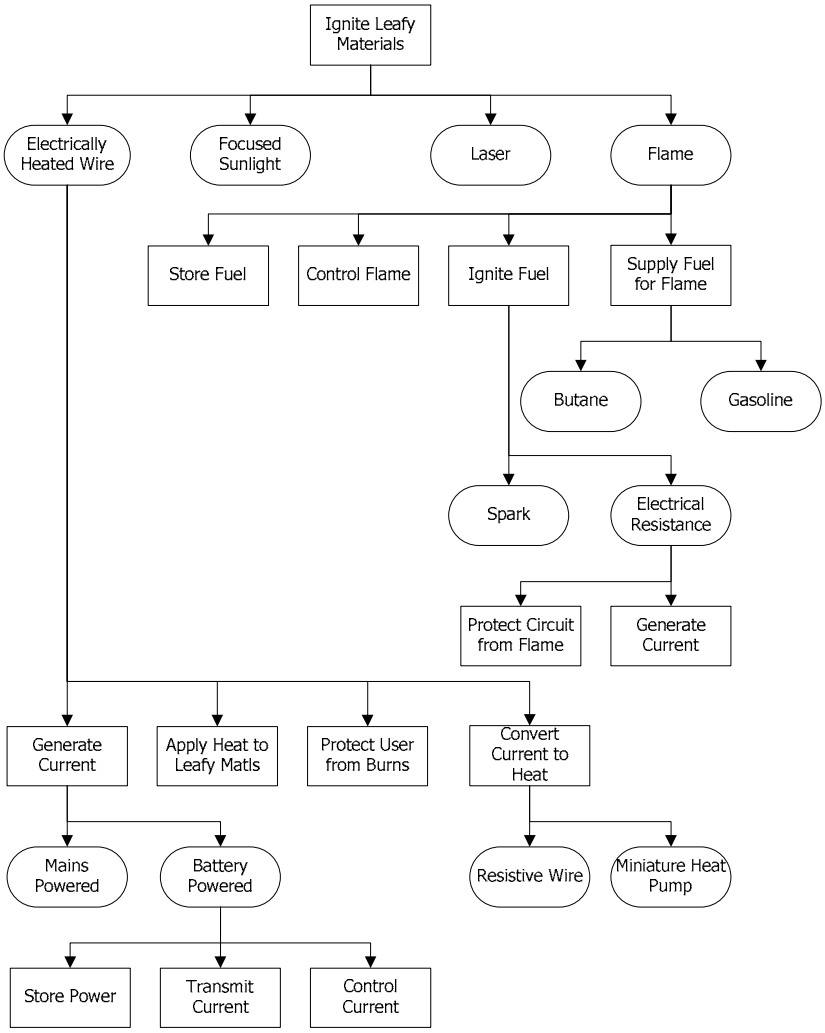


Figure 2.6 Function/Means diagram for a cigarette lighter (Salustri, 2000).

2.1.4 A Critique of Three Design Approaches

All of three approaches in Sections 2.1.1, 2.1.2 and 2.1.3 have benefits and drawbacks. When the bottom-up approach deals with the physical solutions, it is very weak at the point of finding optimum or best designs for complex systems. It is obvious that finding a solution is not the sole concern of today's engineer. Reaching an optimal solution is a necessity rather than an option in today's engineering efforts. Moreover, variations in designs through bottom-up approach are limited to changes in configuration, parameter or

components. Generation of creative ideas requires differences beyond physical alterations.

However, as the top-down approach proposes a powerful strategy to reach the optimal or best functional topologies, it may become very time consuming and sometimes generated solutions can not be physically realizable. This approach applies to the designer's experience, foreseeing ability or trial and error procedure in a highly combinatoric domain of detail. This sometimes causes the top-down design methodology to be a fruitless effort.

Although combined approach assures the physical realization of the artifact, it requires decisions on form at the very beginning steps of the design process. Therefore, inexperienced designers using this strategy may get stuck into the same drawbacks as bottom-up approach.

Consequently, a top-down design strengthened with a functional reasoning technique or an intelligent search strategy seeking solutions with a high probability of success is the most promising approach in solution of design problems. Moreover, employment of a graph based representation scheme with a common vocabulary (e.g. reconciled functional basis) has advantages on standardization of this approach.

2.2 BEHAVIORAL MODELING

Whichever the direction of design activity (i.e. function to form or form to function), an evaluation whether the initial requirements are met is required. A cheap, fast and reliable way of this evaluation is making a virtual prototype of the designed system or product by using behavioral modeling techniques. Behavioral modeling and simulation is a very broad area including all engineering disciplines. However, this section is limited to classification of modeling paradigms and languages and identification of issues that are particularly important in support of multi-technical product design.

According to the level of design, a detailed simulation of the system is not always required. At the beginning stages, coarse but reliable models are very useful for taking feedback of design decisions. The expectations from the model increase as the performance evaluation results become essential for decisions in further levels of design. Figure 2.7 illustrates the position of behavioral modeling in design world.

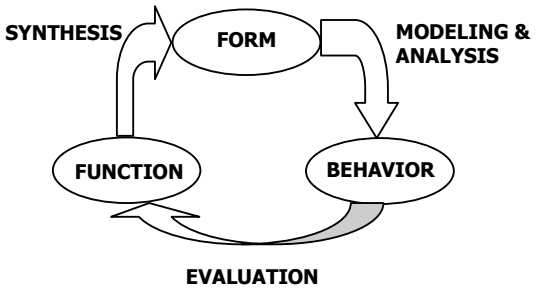


Figure 2.7 Position of behavioral modeling in design procedure (Paredis, 2001).

In order to derive mathematical background of behavioral model, the basic material and energy balance equations are considered. In the literature, there have been many attempts to describe the flow of material and energy in a product. As a result, some commonly accepted modeling paradigms, languages and tools have been developed. These are compared and classified according to the following criteria: graph based versus language based (i.e. textual) paradigms, procedural versus declarative models, multi-domain versus single domain models, continuous versus discrete models, and functional versus object oriented paradigms (Sinha *et al.*, 2001).

Decision on modeling paradigm is the initial step of behavioral modeling. In modeling of systems including parts from different domains or multi-technical products, graph based paradigms such as high level Petri nets, hybrid automata, bond graphs, linear graphs or block diagrams are preferred. These paradigms are briefly explained below.

- Petri Nets are bipartite directed multi-graphs, which are used to model procedures, organizations and devices (systems in general), in which regulated flow of objects and information occurs. Both continuous and discrete systems can be modeled by using Petri Nets (Reisig, 1985; Reisig, 1992).
- Hybrid automata define operations in hybrid systems as a sequence of steps. Within each step the system state evolves continuously according to a dynamical law until a transition occurs. Transitions are instantaneous state changes that separate continuous state evolutions (Alur *et al.*, 1995).
- Bond graphs aim at deriving domain independent description of a system (Broenink, 1999). It is based on energy conserving junctions that connect energy storing or transforming elements through bonds. Although bond graph representation is domain independent, it is difficult to describe a hybrid system consisting of continuous and discrete parts. However, there are some recent studies to develop a hybrid bond graph modeling technique for the systems that combine energy and signal flows (Mosterman and Biswas, 1995).
- Linear graphs interpret systems as a collection of a finite number of energy manipulators regardless of the physical media involved (Platin *et al.*, 1991). Unlike bond graphs, linear graphs reflect the system topology directly. They are domain independent and can be easily extended to hybrid systems (Sinha *et al.*, 2001).

Modeling languages for multi-technical systems are also object oriented and declarative in nature. Unlike procedural languages, declarative languages are equation based. Only state transition equations are established and conversion of these equations to software procedures is left to simulation engine. These languages are domain independent and able to describe hybrid systems as well. Some well known modeling languages with these features are SIDOPS+ (Breunese and Broenink, 1997) a bond graph based language, ASCEND (Piela *et al.*, 1991) employing block diagram based representation, VHDL-AMS

(IEEE, 1999) relying on linear graphs and SynchNet (Ziaei and Agha, 2003) a Petri net based coordination language for distributed objects.

Some other object oriented non-casual physical modeling languages developed within last two decades are ObjectMath (Fritzson *et al.*, 1993), Omola (Mattsson and Andersson, 1992), Smile (Jochum and Kloas, 1994), ULM (UML) (Jeandel *et al.*, 1996) and NMF (Sahlin *et al.*, 1994) and Modelica (Broenink, 1997; Elmqvist and Mattsson, 1997). Modelica is intended to be a superset of the aforementioned languages and to become a neutral exchange format for model representation (Tummescheit *et al.*, 1997). Therefore it allows integration and reuse of model knowledge developed in different modeling and simulation environments.

These modeling languages are more relevant for software developers rather than end users. By implementing these languages, they develop high level modeling and simulation tools to physical system modelers. Only some of these high level modeling and simulation tools are mentioned below.

ClearSim-MultiDomain (Krisp and Müller-Schloer, 2000) has been developed for the timing and functional validation of a virtual prototype. In ClearSim, UML diagrams provide a standard notation in specification of designer's requirements.

20-SIM (Broenink and Kleijin, 1999; Amerogen, J. van, 2000) is another tool to support engineering activities in the processes of design, analysis and diagnosis of multi-domain systems and products. All information about model elements used in 20-SIM has been specified in SIDOPS+ model description language.

Dymola (Elmqvist *et al.*, 1996) is a visual object oriented modeling environment integrated with Modelica. It is used in modeling and simulation of continuous systems.

Schemebuilder (Bracewell and Sharpe, 2004) utilizing the facilities of Dymola/Modelica is another modeling and simulation environment. It supports the designer in both functional and physical design stages. Schemebuilder represents designs as schemes structured based on the Function/Means tree approach and simulations of these designs are performed by the help of a partially bond graph based ontology.

Although there are many other modeling and simulation tools in literature, one of them has found a wide application area especially in control system designs for multi-domain hybrid artifacts namely MATLAB/SIMULINK. SIMULINK is a block diagram based, object oriented, declarative modeling environment. It provides designer with a visual interface both for SIMULINK libraries and MATLAB functions (Mathworks, 2004c).

All of these modeling languages and tools compare designs in terms of performance evaluations, which require mathematical descriptions of energy and material flows included by the artifact. These flows become mathematically identifiable at concrete levels of design process. However, at the initial stages, if top-down approach is adopted, carrying out performance evaluations is very difficult. Therefore in order to model and simulate designs at preliminary phases, a theoretical framework built on Petri Net theory, PNDN (Petri Net Based Design Inference Network) and a simulation software, DNS (Design Network Simulator) have been developed. PNDN (Erden, 1999; Korkmazel, 2001) is intended to represent and analyze artifact behaviors and logical relations that arises these behaviors through information flow. This study utilizes information flow to compare and reveal flaws and drawbacks of examined design alternatives. Proposed theory was implemented in computer environment by Güroğlu (1999) and Coşkun (2004).

Construction of a virtual prototype of an artifact is the main aim of modeling and simulation attempts. A virtual prototype parades all behaviors of the artifact in its real working environment. Although it is a powerful tool for appraisal of designs, other evaluation metrics are also required to guide the

design activity before completion of the design process. These metrics are involved especially at concept generation phase. They may be related to cost, complexity, probability of success or another objective measure. Some of these metrics are overviewed in Section 2.4.

2.3 REASONING TECHNIQUES AND EVOLUTIONARY SEARCH STRATEGIES IN DESIGN

Automation of conceptual design involves generation of solutions for design tasks without need of human supervision. This can be achieved by employing either a reasoning technique or an evolutionary search strategy in the solution domain. In the literature, there are significant examples for both of these approaches. Some of them are discussed in following sections.

2.3.1 Functional Reasoning in Design

Reasoning at functional level to solve a design problem can be performed in two ways namely descriptive or prescriptive. Descriptive way aims at imitating human way of thinking. It tries to describe the nature of the human concept generation. Prescriptive way focuses on generation of a reasoning scheme to constitute solution alternatives. Since the former is mainly associated with cognitive sciences, the studies on the latter are mentioned here.

Chakrabarti and Bligh (2001) cited three influential functional reasoning techniques and presented a new approach in their study. The earliest one was proposed by Freeman and Newell (1971). This study operates a similar procedure used in function/means tree construction. All functional requirements are mapped to some structures (i.e. physical entities) and these structures bring new required functions. This process continues until all the functions required are provided by some structures. The resulting combination of these structures provides the designer with a solution to the design problem. However, if no structure is found to partially or completely fulfill a

function at a given level, this methodology can not offer any solution to the designer.

This limitation is dealt with in Yoshikawa's approach (1981; 1985). This approach bases on modification of an existing solution. After describing design problem in terms of functional requirements, the design process starts with a temporary solution. This temporary solution is evaluated by comparing the original functional requirements and the functions provided by the temporary solution. The aim of designer would reduce the difference. For this purpose, wrong components in temporary solution are identified and replaced by other components having more potential to contribute to satisfaction of the problem. This process is continued until the solution becomes satisfactory. This reasoning technique has some underlying assumptions. At first, the functions provided by the temporary solution in the absence of wrong components could be identified. This assumption is defined as deducibility. The second assumption is availability of a satisfaction evaluation criterion (i.e. evaluatability). This criterion is employed to figure out whether the potential of the solution to solve the problem increases after modification or not. The third assumption, monotonicity, states that it should be possible to monotonically modify the temporary solution. In other words, increase in the satisfaction of requirements should ensure a move towards a valid and complete solution. This can only be possible with the last assumption. The last assumption, called decomposability, involves the artifact comprising independent clusters of components. This provides the designer with the ability of separately control various functions. In short, replacement of a cluster should not affect the functionality of the rest of the solution. Even though the temporary solution validates all of these assumptions, there are still a vast number of wrong components to be checked against the satisfaction of evaluation criterion. Due to these limitations imposed by assumptions and computational intensity, this technique does not look promising in design of complex artifacts.

Another widely accepted reasoning technique was proposed by Pahl and Beitz (1988). This technique carries out the procedure of functional design tree construction to express the problem in a solution neutral way. It decomposes the overall design task into sub-functions until reaching sufficiently simple function definitions. Then alternative physical concepts are sought for this functional structure. After the evaluation of the alternatives, the best one is chosen. Chakrabarti and Bligh (2001) indicated the need of this technique for a finite, distinct and complete function set as a drawback. Certainly the predetermined function set should comprise a finite number of functions (i.e. a finite function set). There should be no overlapping function definitions in it (i.e. a distinct function set) and finally all solutions should be expressed by the combination of these functions (i.e. a complete function set). The impossibility of generation hierarchical function structures in a solution neutral way (i.e. law of vertical causality) was pointed out as well. In addition, Chakrabarti and Bligh (2001) claim that in this approach, it is possible to find solution concepts only if each atomic function corresponds to a component-solution. If a function structure rather than a single function corresponds to a component-solution, it can not be reasoned. This difficulty was addressed as the problem of partitioning in their study.

Due to these limitations of Pahl and Beitz's approach (1988), Chakrabarti and Bligh (2001) have proposed a new reasoning technique in their study. In this technique, a recursive problem definition is used. In each step only a part of overall functional requirement is handled and a set of alternative solutions is synthesized to satisfy this part. After the evaluation of this partial solution, overall problem definition is revised and another part of requirements is handled in the next step. This process continues until satisfaction of all requirements. Then found physical solutions are aggregated to constitute the complete solution for the problem.

Another reasoning technique was discussed by Sharpe and Bracewell (1995). They used bond graph reasoning in construction of function/means tree of a product. This approach provides a set of rules for the decomposition of

energetic systems. These rules are based on the fact that only similar energy ports must be connected to each other. Therefore, the correct components should be selected to be able to propagate energy variables. This selection limits the set of functions used in tree construction. This reasoning technique is only applicable to the functions having an effect on energy flow. It does not offer any methodology to identify functions operating on information or material.

An experience derived heuristics can also be used as a reasoning technique in design (Potter *et al.*, 2003). Heuristic knowledge comprises "rules of thumb" concerning the manner in which the inference process is driven. It is derived by case based reasoning making use of previous design experiences. This knowledge base generally comprises if-then rules.

All of these approaches except Pahl and Beitz's approach (1988) can not be adopted in creative design activities due to their dependency on physical solutions to proceed. The solution neutral way of Pahl and Beitz's approach (1988) allows generation of both creative and innovative designs. The questions on the necessary features of predetermined function set can be removed by the use of a detailed functional vocabulary (e.g. reconciled functional basis). Furthermore, if graph based representation is preferred instead of hierarchical representation, there will be no need to operate the law of vertical causality. Consequently solution neutrality will be kept. Thanks to atomic function and flow descriptions in the vocabulary, the designs are naturally generated at a certain level of abstraction. Moreover, above mentioned partitioning problem can be defeated by the identification of modules. Ulrich and Tung (1991) define modules as physical structures that have a one-to-one correspondence with functional structures. A module identification heuristic in functional domain would overcome this difficulty (Stone *et al.*, 2000).

2.3.2 Evolutionary Search Strategies in Design

In late 1950s, researchers have paid their attention to the use of evolution as an optimization tool for engineering systems. This optimization tool used the operators inspired by genetic variation and natural selection. In the following years, evolutionary computation including evolutionary strategies, evolutionary programming and genetic algorithms has become an active area of research.

Detailed theory of Genetic Algorithms (GA) was pioneered by Holland (1975). Holland's primarily aim was not to develop an algorithm for optimization issues but rather to study the phenomenon of adaptation as it occurs in nature (Mitchell, 1996). Due to the fixed length encoding of genetic algorithms, this technique has been limited to problems whose topology remain fixed, but whose parametric values can be changed (Roston, 1994). Some example application areas include controller parameter estimation (Downing *et al.*, 1996), VLSI circuit layout design, filter design, financial applications, scheduling, shape and structural design, truss optimization (Burton, 2004) etc. In order to remove this fixed topology limitation and improve GA performance, Goldberg *et al.* (1989c) has proposed the notion of messy GA. Messy GA makes it possible to work with variable length chromosomes and expands the application areas of GAs.

The idea of application of GAs to a population of computer algorithms to design a new algorithm automatically was introduced by Koza (1992; 1994). Besides self-reproduction of software and self-programming, this new technique namely Genetic Programming (GP) has found a wide application area for itself such as data mining, task prioritization, path planning, distributed problem solving, natural language processing, symbolic regression and network routing etc.

In addition to these application areas of evolutionary computation, both GA and GP have become strong tools of engineers in solving engineering design

and optimization problems. Some examples in the literature are mentioned below.

In the studies of Sims (1994a; 1994b) evolutionary strategies have been applied to morphological design issues. In a part of these studies, different creature morphologies are generated and the generated morphologies are evaluated according to their performances in realizing different behaviors such as swimming, walking, jumping and following. For the description of morphologies, a graph based genetic language using nodes and connections as its primitive elements was developed. In the remaining part of these studies, control systems for generated creatures were also determined by using evolutionary strategies.

Another morphological design application has been realized by Pollack *et al.* (2000). At the beginning, this study handled morphology as the arbitrary networks of linear actuators and bars. It employed evolution to generate sufficiently proficient structures from these networks. As a case study, some structures were emerged to carry out the task of locomotion. The distances traveled by the generated structures are defined as evaluation criterion. It has been observed that the evolved robots exhibited various methods of locomotion, including crawling, ratcheting and some forms of pedalism.

An alternative application on design of locomotion system has been presented by Roston (1994). In his study, a hybrid evolutionary technique (GA/GP) was developed and applied to a configuration design issue of a frame-walking robot. It has been observed that the developed methodology provided promising solutions in configuration design problems.

A group of evolutionary applications has focused on the modularity issue in design. Koza (1994) has proposed a promising method relying on the description of partial solutions in the form of reusable blocks. These solutions called Automatically Defined Functions (ADFs) decrease the depth of trees involved in genetic programming and consequently the computation load.

Unlike Koza's approach (1994), Lipson *et al.* (2001) have employed an evolutionary mechanism to generate modular designs without module descriptions. This mechanism has based on the observation that modular designs have higher adaptability and better survival rates under changing requirements.

In addition to these evolutionary design applications, there exist some others even at molecular level in the literature. In particular, these studies concentrate on pharmacological problems such as drug design (Lawton and Wipke, 1999) or protein folding (Schulze-Kremer, 1996).

All of these studies indicate that the majority of evolutionary design applications focus on the physical design issues such as component selection or configuration design. However, there is no study that carried out evolutionary techniques to design a functional topology. Such a study with a correct evaluation measure would eliminate the need for a functional reasoning technique. Some evaluation measures mentioned in the literature for functional designs are presented in following section.

2.4 SURVEY ON EVALUATION METRICS FOR FUNCTIONAL DESIGNS

Evaluation of artifacts at functional level is a very difficult task to perform due to the abstract nature of functional design process. At this level, only working principles are determined and no other information is available for performance calculations. However, some characteristics of function structures can be used for comparison. First of all, it is preferable to construct artifacts using independent functions. Therefore dependency of functions can be considered as a comparison criterion for generated design alternatives. The number of functions in an artifact can be employed as another measure for comparison. Designs should perform their tasks through operating minimum number of functions. Finally, all functional structures in a design could be physically achievable. The last characteristic is also a necessity to accept a

function cluster as a design alternative. These characteristics of function structures provide designer with a limited evaluation capability and they can not be used as general evaluation metrics. According to these characteristics, it is possible to generate equally acceptable solutions from the functional point of view. Some evaluation metrics employed in the literature are information content, value analysis and complexity. These are briefly explained below.

Information content determines the probability of success of a design alternative (Suh, 1990). Design with minimum information content has highest probability of success. The information content "I" associated with the probability is defined as

$$I = -\log_2 p \tag{2.1}$$

The unit of information content is bits. Artifacts including many functions should satisfy all of them at the same time. The logarithmic function makes information content additive. Then the information content of an artifact is

$$I = \sum \left[\log \frac{1}{p_i} \right] \tag{2.2}$$

where p_i is the probability of success of each function. Minimum information content states that generated alternative requires the least amount of information to achieve the design goals. When probabilities of all functions included by the artifact are equal to unity, the information content becomes zero. Then design becomes completely achievable. Conversely, the information required is infinite when one or more probabilities are equal to zero. This shows that design is unreachable.

In the real world, the probability of success is provided by the intersection of the tolerance defined by the designer to satisfy the function and the tolerance

of the system to produce the part within the specified tolerance. This intersection is illustrated in Figure 2.8.

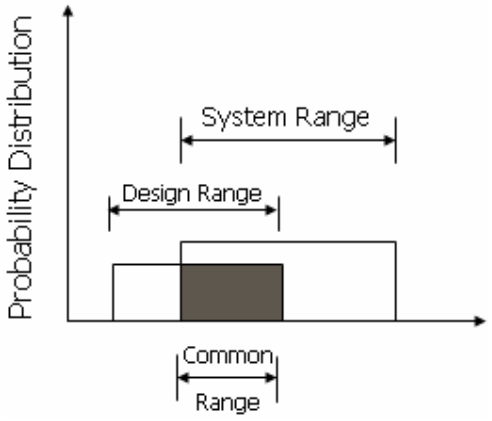


Figure 2.8 Intersection of the design range and system range (Amos *et al.*, 2001).

$$I = \log_2\left(\frac{SystemRange}{CommonRange}\right) \tag{2.3}$$

One disadvantage of this metric is its need for decisions in physical domain. In order to calculate information content of an artifact, its physical design should also be completed.

Another evaluation metric, value analysis aims at achieving the total function for lowest overall cost (Miles, 1972). Value analysis decomposes artifacts in functional domain, and establishes means for each function. According to these means, an approximate cost is assigned to each function. By considering the interactions between functions, these values are added and the cost of achieving the overall task is determined. It is obvious that this technique also needs the completion of the decisions in physical domain.

Finally, complexity can be used as an evaluation criterion of artifacts. Summers and Shah (2003) give six different definitions of complexity in their studies. These definitions vary from design problem complexity, design and manufacturing process complexity to design artifact complexity. In the

literature, numerous measuring methods are proposed for these complexity definitions. These methods can be grouped as: computational views of complexity, information based views of complexity and traditional design views of complexity. One of the information based measuring methods looks promising to measure the complexity of artifacts generated completely in functional domain. This method calculates structural complexity through considering functions and relationships constructing the artifact. A detailed explanation for this measure will be presented in Chapter IV.

CHAPTER III

AN APPRAISAL OF EVOLUTIONARY METHODS

There are numerous well established search and optimization techniques in the literature. Some of these techniques are illustrated in Figure 3.1. Enumerative methods, a type of deterministic search techniques, in principal search systematically every possible solution one at a time. Therefore, they require vast amount of time and calculation power in large search spaces. Meanwhile, another deterministic search technique, traditional calculus based method needs continuous mathematical functions and their derivatives, however, it is not always possible to define the problem in the form of a continuous mathematical function. To overcome these difficulties, stochastic techniques are suggested.

Stochastic techniques determine the next search point with a probability obtained from the search information so far. For instance, simulated annealing looks for a good solution to an optimization problem by trying random variations of the current solution. A worse variation is accepted as the new solution with a probability that decreases as the computation proceeds. This technique chooses the new solution if the probability of the new solution is greater than the probability of the current solution (Zhang and Kim, 2000). Similarly, evolutionary strategies, a type of evolutionary algorithms, basically creates a new solution by adding random noise to current solution. If the new solution is better, search proceeds utilizing the new solution, if not the older solution is retained (Langdon and Qureshi, 1995). In other words, these stochastic techniques use past experiences to guide future events. This

characteristic causes a considerable decrease in the need of calculation time and power.

All evolutionary algorithms are based on Darwin's Natural Selection theory of evolution, where a population is progressively improved by selectively discarding the worse and breeding new children from the better (Langdon and Qureshi, 1995). Koza (1992) states the conditions, on which the evolutionary process in nature depends, as;

- An entity should be capable of reproducing itself.
- Such self-reproducing entities should constitute a population.
- There should be some variety among the self-reproducing entities.
- This variety should be associated with rate of survival and the reproduction of the entity in its environment.

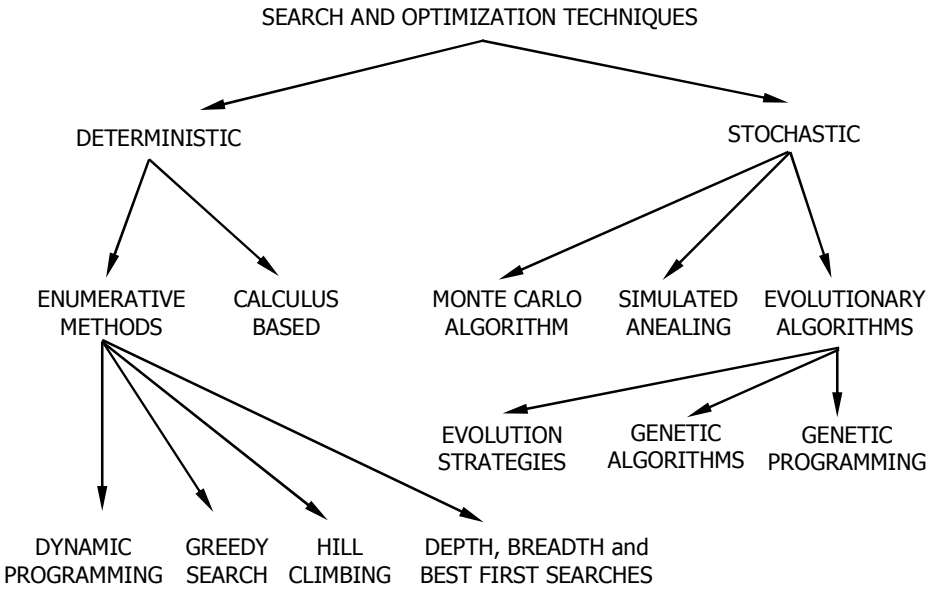


Figure 3.1 Search and optimization techniques.

Therefore selective replication of individuals, information exchange among them and inheritance are also the key concepts of evolutionary search and optimization techniques. Almost all evolutionary techniques operate the procedure presented in Figure 3.2.

Evolutionary techniques have some merits and shortcomings. Primarily these techniques have the advantage to create new individuals without requiring understanding of procedures used to generate them. This feature makes the strategy domain independent and computer implementable. In addition, randomness in genetic operations increases the applicability of these techniques to complex problems. Therefore it becomes possible to extract many successful and interesting solution suggestions, some of which would be difficult to invent or build by traditional approaches (Sims, 1994a).

```
procedure Evolutionary Algorithm;
        begin
            initialize population P_{t=0};
            evaluate all individuals in P_{t=0};
            do
            t=t+1;
            select P_t from P_{t-1};
            recombine P_t;
            evaluate P_t;
            while(termination criterion not satisfied)
        end.
```

Figure 3.2 A general algorithm for evolutionary techniques.

However, there are still some poorly understood factors that affect the success of these techniques. These include ambiguity in the choice of initial populations, need for experiments in determination of constraint handling techniques and case dependency of selection and replacement mechanisms.

While evolutionary techniques keep a part of search history by inheritance, random mechanisms of genetic operations add randomness to them. Therefore evolutionary techniques provide a crossover of random and informed search strategies.

Evolutionary techniques differ in representation scheme. While genetic algorithms (GA) prefer string based representation scheme, genetic programming (GP) utilizes trees to represent the individuals. The preferred representation scheme has to be general enough to cover all possible

solutions. It also has to be suitable for computer implementation. In Sections 3.1 and 3.2, an overview for genetic algorithms and genetic programming are given respectively.

3.1 INTRODUCTION TO GENETIC ALGORITHMS (GA)

Genetic algorithm defines the individuals with the string representations (usually fixed length representation is preferred). For this purpose, at first, a representation scheme for the solution of the problem is determined. Representation scheme is generated through the coding of the parameters. Genetic algorithm works with this coding rather than the parameters themselves. Specification of the representation scheme requires selection of the string length and the alphabet size. Bit-string representation of the problem is the most preferred representation type in genetic algorithms. In this representation, the alphabet is binary. Non-binary representations are also possible but tend to be more complex without yielding any additional benefit (Roston, 1994).

In string based representation, a part of string describing a property is named as a gene. The values taken by this gene are alleles. All values that can be taken in the problem should be described by these alleles. Strings containing more than one gene as shown in Figure 3.3 are called the chromosomes. This chromosome is analogous to the base-4 chromosomes present in our DNA. In GA community, the haploid model (i.e. one-chromosome individuals) is used. However, diploid models have also been used in the past (Goldberg, 1989a).

Figure 3.3 A chromosome, which comprises 3 genes.

After the identification of the scheme, the next step is to define the size of population. Population sizing is a trade off between reaching solutions of a

certain quality and increasing computational costs. While larger populations result in better solutions, dealing with unnecessary individuals causes wasting time and computational resources. In literature, there exists small number of theoretical studies on determining the adequate population sizes (Goldberg, 1989b; Goldberg *et al.*, 1992; Harik *et al.*, 1999). Many genetic algorithm applications prefer sizing their populations empirically in general.

After finding the population size, the individuals constituting the initial population should be defined. The individuals may be constructed randomly or a predetermined set of individuals may also be defined as an initial population.

The next step is to determine a fitness measure for genetic algorithm. By using the fitness measure, each possible string in the search space is evaluated and a fitness value is assigned. The fitness measure is often inherent in the problem (Koza, 1992). For example, if the aim is to find the object, which has the largest area in the search space, the fitness measure is the multiplication of the length and the width of the object, which are represented by two genes of a chromosome.

Genetic algorithm transforms the individuals of initial population each with an associated fitness value, into a new population (i.e. the next generation) using reproduction operation. Reproduction operation determines the fitter individuals in the current population and copies these individuals without change into the next generation. Therefore, fitter individuals in the population survive and others become extinct. There are number of selection methods used to perform reproduction operation such as fitness proportionate selection, linear ranking selection and truncation selection etc. (Blickle and Thiele, 1995). The most common selection method in literature is the "fitness proportionate reproduction". This method copies the individuals in the current generation into the next generation with a probability proportional to their fitness.

The next step after reproduction is the modification of the individuals of the new population. This step comprises two genetic operations namely, crossover and mutation. The crossover operation aims to generate new individuals by crossing the individuals. Therefore the fittest genes of individuals are transmitted to their descendants through sexual recombination. Crossover operation starts with two parents and ends with two new offspring. Parents are selected proportionate to their fitness values. Crossover probability (p_c) indicates the percentage of the population, which will participate in crossover operation. Crossover points on participating members are determined randomly. Then parent individuals are crossed at that point and two offspring occur in the new population. This operation is illustrated in Figure 3.4.

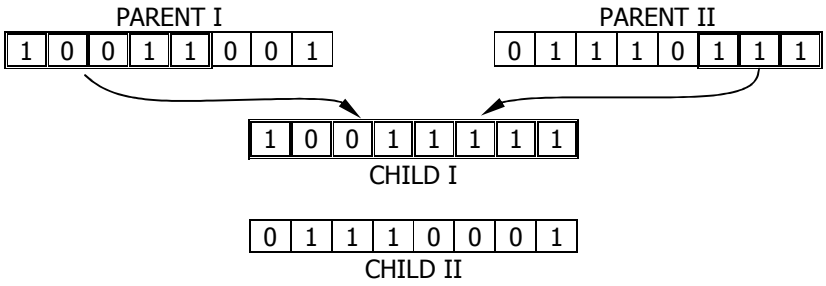


Figure 3.4 Crossover operation in genetic algorithms.

Another individual modification operation is mutation. Frequency of mutation operation is controlled by a parameter called mutation probability (p_m). Mutation operates on a single individual. This operation is especially needed for increasing the genetic diversity of the population. In some cases, in order to avoid from getting stuck into a local minima, occurrence of some bits in string could become extinct in the earlier generations of the population. These bits can be needed to reach global optimum. Therefore mutation operation can also be used to remind the extinct bits to the population. Mutation is a secondary operation. Therefore it is not expected to obtain a global optimum via a single mutation. However it is a strong tool that provides a way to restore the genetic diversity lost in the generations of the population.

In binary string representation, mutation operation is applied by converting the value of the bit at the point of mutation. Mutation operation can be applied more than one point in a string. Figure 3.5 illustrates a sample mutation operation.

Figure 3.5 Mutation operation in genetic algorithms.

Finally, run of a genetic algorithm is terminated in two ways; first, if the fitness of the best individual in the run is close to the optimal solution with an acceptable predefined error value, operation of the algorithm is terminated. The second way of termination is the execution of the predefined maximum number of generation. Although an acceptable solution can not be found, operation is terminated.

Below, a genetic algorithm implemented in MATLAB environment demonstrates the capability of genetic algorithm to avoid local minima in searching for global minimum. For this purpose, Rastrigin's function, which is often used to test the genetic algorithms, is employed. Its many local minima make it difficult for standard, gradient-based methods to find the global minimum (Mathworks, 2004a). Figure 3.6 illustrates the plot of Rastrigin's function shown in Equation 3.1.

$$Z(x_1, x_2) = 20 + {x_1}^2 + {x_2}^2 - 10 \cdot (\cos 2\pi x_1 + \cos 2\pi x_2) \qquad (3.1)$$

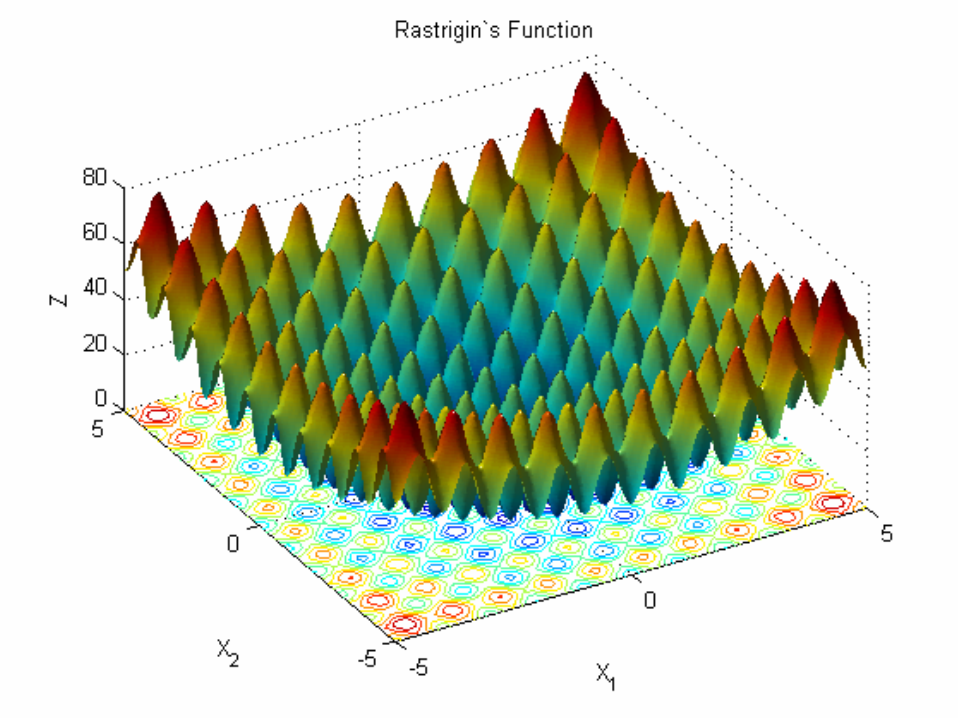


Figure 3.6 Plot of Rastrigin's function.

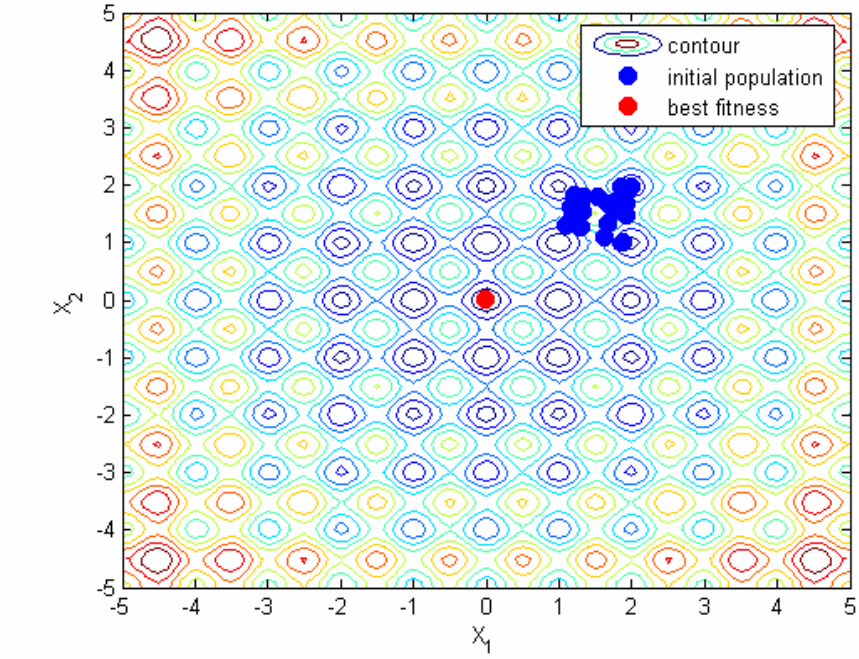


Figure 3.7 Contour plot of Rastrigin's function with initial population and best individuals.

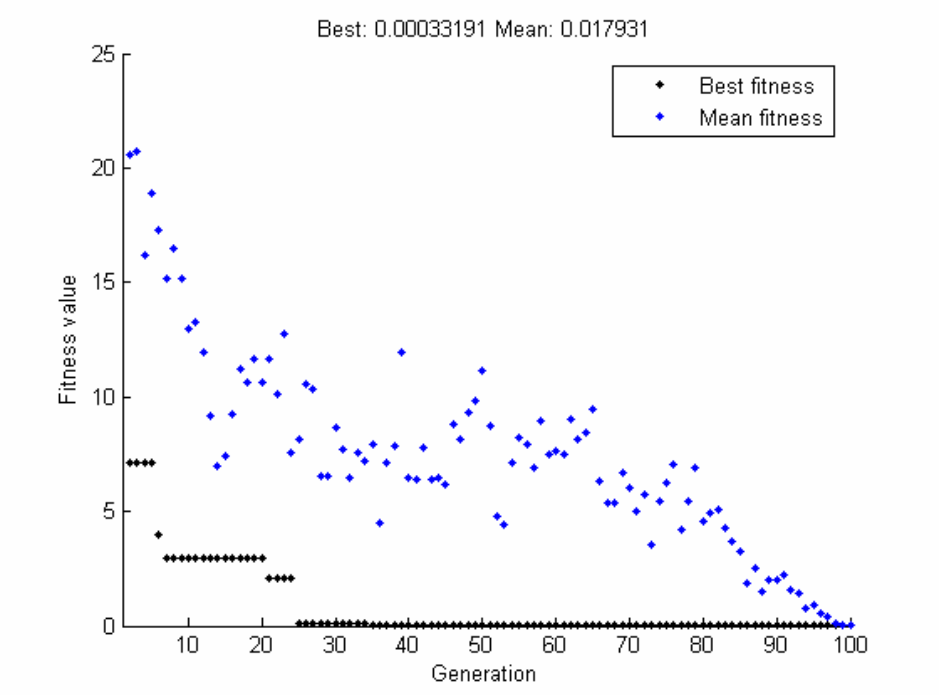


Figure 3.8 Best and mean fitness values of implemented genetic algorithm.

Population size and maximum number of generations for genetic algorithm are set as 25 and 100 respectively. All individuals at initial population, blue circles in Figure 3.7, are constructed randomly. The genetic algorithm successfully reaches the global optimum at 25th generation indicated by the red circle in Figure 3.7. All individuals in the population also converges the global optimum at nearly 97th generation, shown in Figure 3.8. This figure also shows the best and mean fitness plots of population for all generations.

3.2 INTRODUCTION TO GENETIC PROGRAMMING (GP)

Genetic programming is a special application of genetic algorithm theory to automated programming concept. The main aim is to develop a method, which teaches the computers to solve problems without needing explicitly programming. The solutions of the many problems need hierarchical computer programs rather than fixed length character strings. Hierarchical depths of these programs are not known in advance. Therefore it is very difficult to

represent computer programs of dynamically varying sizes and shapes with fixed length character strings.

GP uses rooted trees to represent the individuals in the population. These rooted trees consist of terminals and functions appropriate to the problem domain. As the functions can be standard arithmetic operators, logical functions or domain specific function definitions etc., the terminals can consist of either variable atoms (representing perhaps the inputs, sensors, detectors or state variables of the system) or constant atoms such as the numbers or the Boolean constant NIL) (Koza, 1992). Rooted tree representation of function x^2+2x+3 is represented in Figure 3.9. In this example the function and terminal sets consist of {+,*} and {2, 3, X} respectively.

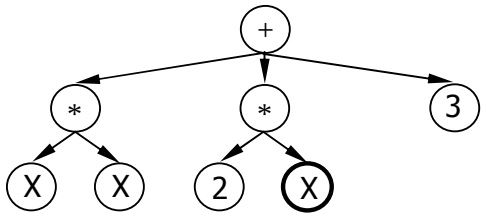


Figure 3.9 Rooted tree representation of x^2+2x+3

Hierarchical representation has two main properties namely closure and sufficiency. The closure property states that each function defined in the function set must be able to accept, as its arguments, all of the possible return values of other functions and all possible terminal values defined in the terminal set. In some cases, such as requiring combined function sets including the arithmetic operators and logical operators at the same time, there may be difficulties in the application of this property. By defining some syntactic rules, these difficulties can be overcome. The closure property is desirable, but it is not absolutely required. If it is not succeeded, alternative strategies such as elimination of the infeasible individuals or application of a penalty to individuals that generate unacceptable results can be applied. The other property, sufficiency, states that the set of terminals and the set of

functions should be capable of expressing a solution to the problem. This property is a precondition of solving a problem with genetic programming.

In order to be able to operate the algorithm shown in Figure 3.2, an initial population of rooted trees is generated considering the properties mentioned above. Similar to the genetic algorithms, the next step is the definition of a fitness measure to assign a scalar fitness value to each individual in the population. These fitness values are employed to perform genetic operations. Genetic programming has two primary (i.e. reproduction, crossover) and five secondary (i.e. mutation, permutation, editing, encapsulation and decimation) genetic operations.

Both the reproduction and the crossover operations obey the same rules as the operations in genetic algorithms. However, the only change occurs at crossover operation in shape. In this operation, different sized parents can be selected. A random point in each parent is chosen as a crossover point. Crossover fragments of the parents are rooted sub-trees, which accept crossover point as the root. The first offspring is produced by deleting the crossover fragment of the first parent and then inserting the crossover fragment of the second parent at the crossover point of the first parent. The second offspring is produced in symmetric manner. This operation is illustrated in Figure 3.10.

In secondary operations, the most common one is mutation operation. Mutation operation starts with random selection of the mutation point in individual. A terminal or a function node can be selected as a mutation point. Mutation operation removes the mutation point and the tree below this point and inserts a randomly generated sub-tree at that point. Figure 3.11 gives an example of mutation operation.

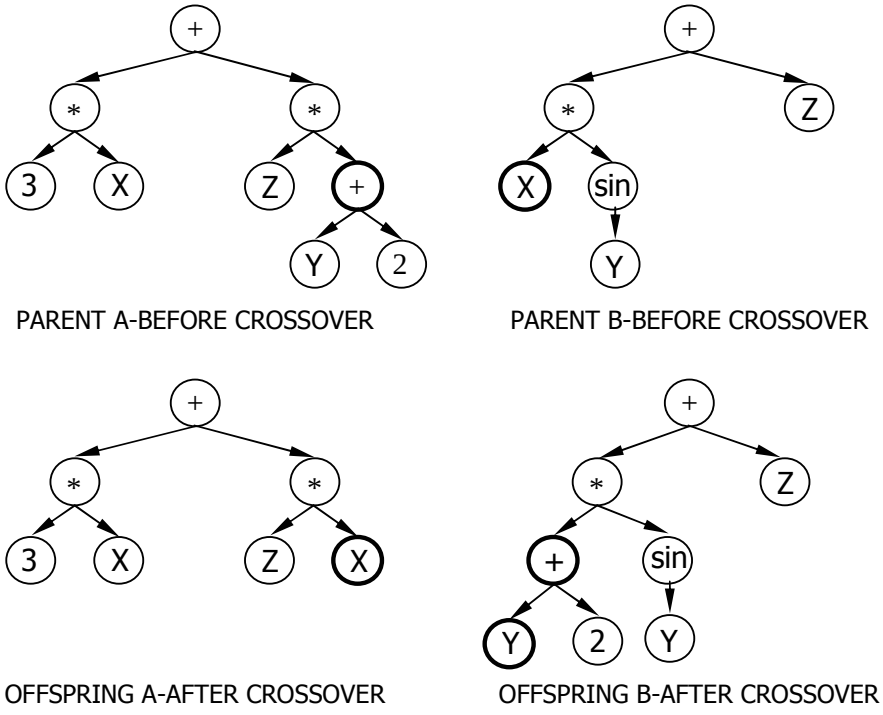


Figure 3.10 Crossover operation in genetic programming

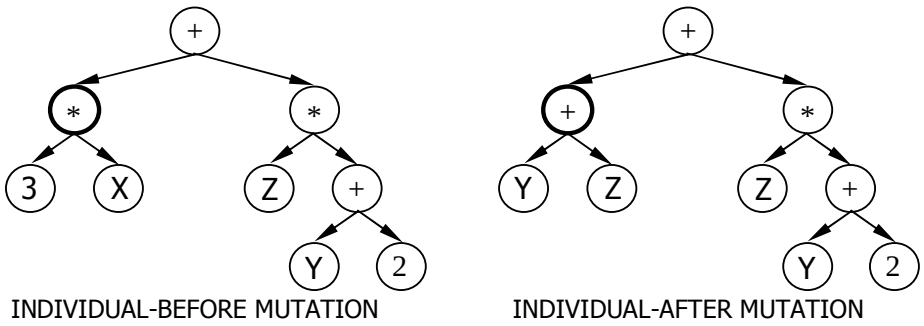


Figure 3.11 Mutation operation in genetic programming

These operations are iteratively performed until reaching termination criterion. As in the case of genetic algorithms, genetic programming also uses a predefined error value and the maximum number of generations as termination criteria.

In order to demonstrate problem solving capability of genetic programming, a regression problem is solved at MATLAB environment through the help of GPLab, which is a third parity genetic programming toolbox for MATLAB (Silva,

2004). At first a regression problem is solved for Equation 3.2 with the data between -3 and +3.

$$x^3 + 2x^2 + 3x - 8 \qquad (3.2)$$

Population size and maximum number of generations for genetic programming are set as 50 and 25 respectively. All individuals at initial population are constructed randomly using the function and terminal sets of {plus, minus, times} and {1, X}. Genetic programming exactly reaches Equation 3.2 at 23rd generation. The best individual and the approximation progress in generations are given in Figures 3.12 and 3.13 respectively.

At the bottom of the tree shown in Figure 3.12, the bordered region performs a fruitless operation. This proves that, GP tends to produce programs that are robust to genetic manipulation, similar in a sense to biological genomes that produce viable individuals even though parts of the DNA sequence might be corrupt (Roston, 1994).

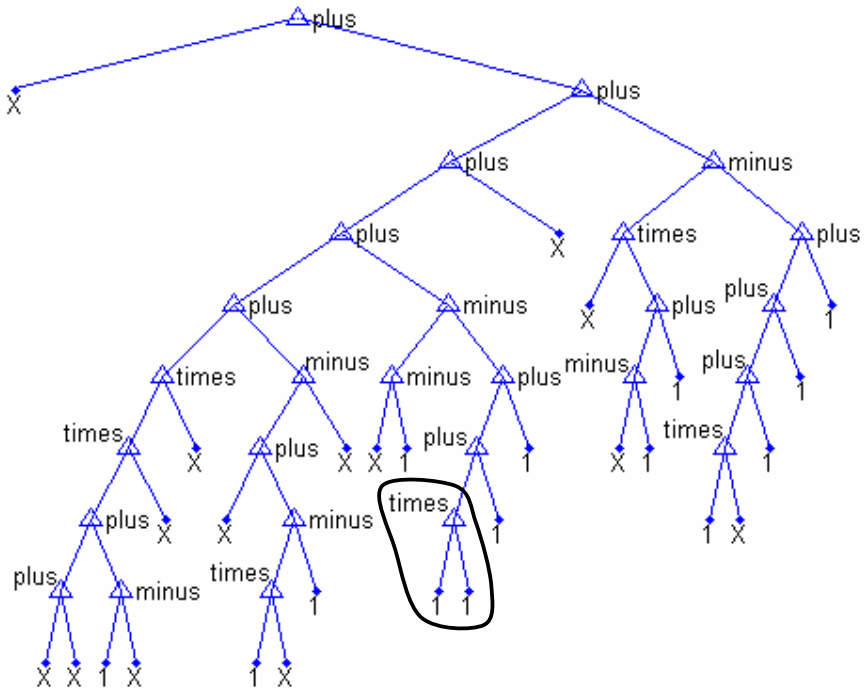


Figure 3.12 Best individual obtained for the regression problem.

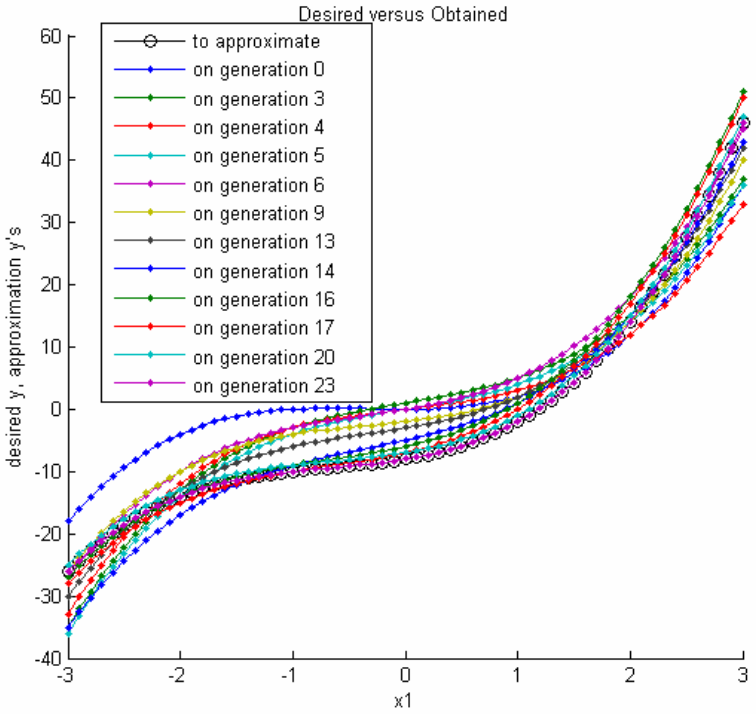


Figure 3.13 Approximation progress for the regression problem.

Randomness gives robustness to evolutionary algorithms. Therefore evolutionary algorithms become capable of handling uncertainties in problems. This can be validated by solving aforementioned regression problem by adding random noise to its data. In this problem, a polynomial will be fitted to randomly disturbed data shown in Figure 3.14.

The parameters of genetic programming are again set to 25 and 50 for maximum number of generations and population size respectively. It is observed that, genetic programming successfully reaches Equation 3.2 at 24th generation. The rooted tree representation of the best individual is illustrated in Figure 3.15. The approximation progress of evolutionary process in generations is given in Figure 3.16. This simple example demonstrates uncertainty handling capability of evolutionary algorithms in the presence of noise.

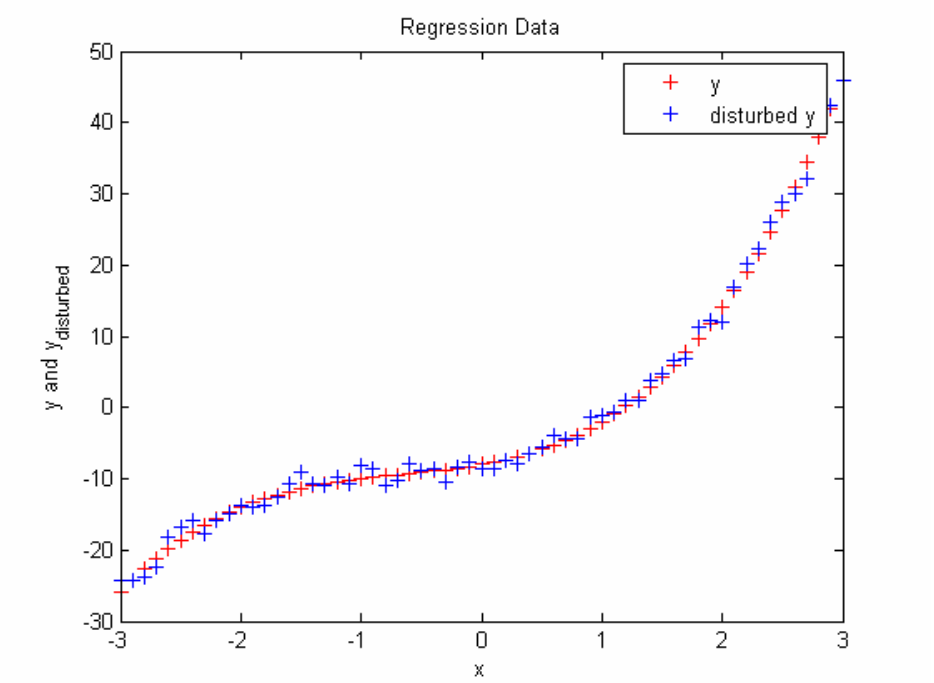


Figure 3.14 Noisy data for the regression problem.

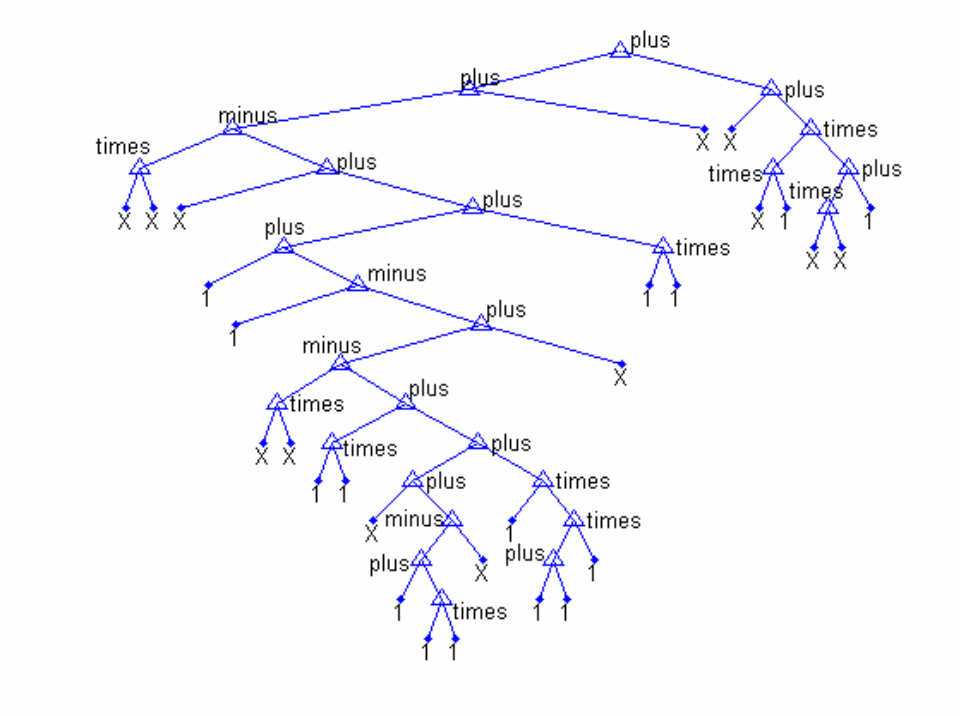


Figure 3.15 Best individual obtained for the regression problem with noisy data.

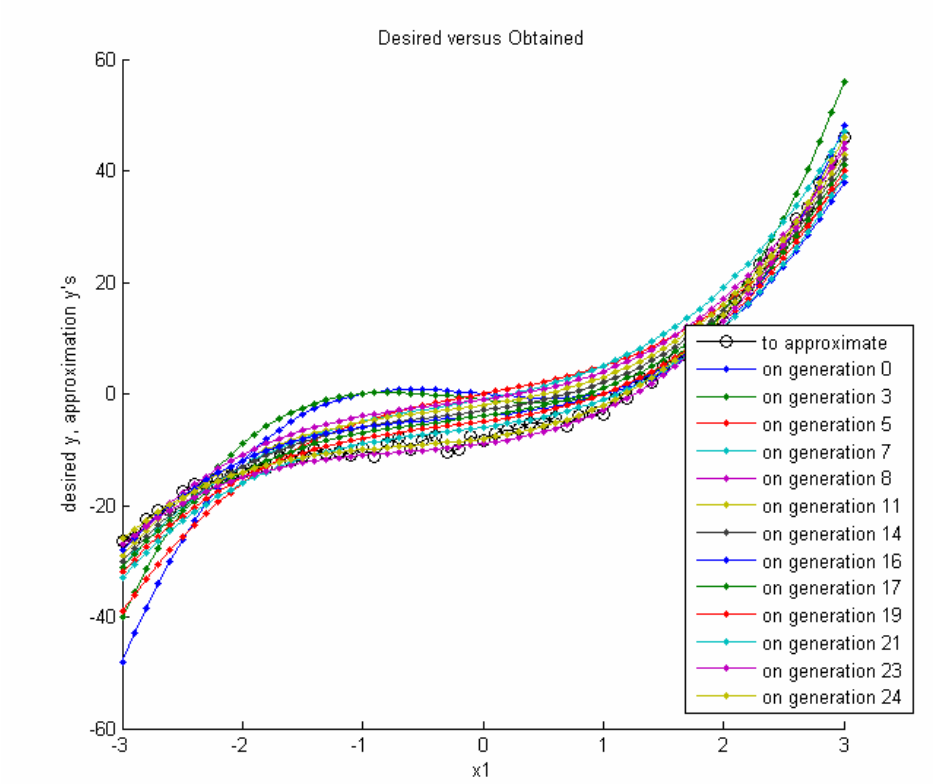


Figure 3.16 Approximation progress for the regression problem with noisy data.

3.3 CONSTRAINT HANDLING TECHNIQUES USED WITH EVOLUTIONARY ALGORITHMS

A vast majority of engineering problems are subject to constraints. Constraint handling affects the performance of all optimization techniques including evolutionary algorithms. Therefore numerous constraint handling techniques are suggested to be used with evolutionary algorithms in the literature (Coello, 2002). These techniques are basically established on two different paradigms (Michalewicz *et al.*, 1996). The first one is based on the penalization of emerged infeasible individuals in the population. Other paradigm aims at maintaining the feasibility of individuals in the population by means of some restrictions and operations.

The most common technique is to use penalties. Basically, penalty is a way of converting constrained optimization problems into unconstrained optimization problems by adding or subtracting a certain value to/from the objective function. This value is generally calculated considering the amount of constraint violation in a certain solution. There are three different views of assigning penalties to infeasible individuals;

- Infeasible individual is penalized regardless of the amount of constraint violation.
- The amount of violation is taken into consideration to calculate the penalty.
- The effort of repairing infeasible individual (i.e. cost of making individual feasible) is considered to find penalty.

Traditionally, weighting of penalties is usually based on experiments. Often the algorithm must be rerun several times before assigning penalties (Carlson and Shonkwiler, 1998). The main problem in determining the penalties is that they are problem dependent. However, it is ideally suggested that the penalty should be kept as low as possible (Coello, 2002). High penalties discourage the exploration of the infeasible region since the very beginning of the search process.

There are several types of penalty functions such as static, dynamic, annealing, adaptive or death penalty etc. (Coello, 2002). Static penalties remain constant throughout the evolutionary process. The number of generations executed is not taken into account in this type of penalty calculations. In some types of static penalties, level of constraint violation is also handled through a distance metric. Applicability of static penalty is limited in highly constrained search spaces. Using a static penalty function, an individual is evaluated by Equation 3.3 (Michalewicz, 1996).

$$fitness(\vec{x}) = f(\vec{x}) + \sum_{i=1}^{m} \left(R_{k,i} \times \max\left[0, g_i(\vec{x})\right]^2 \right) \qquad (3.3)$$

where $R_{k,i}$ are the penalty coefficients used, m is the total number of constraints, $f(\vec{x})$ is the unpenalized objective function and k=1,2, ….,l, where l is the number of levels of violation defined by the designer. $g(\vec{x})$ denotes the function of constraint.

The second category consists of dynamic penalties. This category involves the current number of generation in calculation of the penalty. In general, for minimization problems, penalty function increases as the evolution proceeds. This type of penalty also requires a distance metric to find the amount of violation. It is possible to define dynamic penalty in different forms. One of these forms is presented in Equation 3.4 (Kazarlis and Petridis, 1998).

$$fitness(\vec{x}) = f(\vec{x}) + V(g) \times (A\sum_{i=1}^{m}(\delta_i \cdot w_i \cdot \Phi(d_i(S))) + B) \times \delta_s \qquad (3.4)$$

where A is a severity factor, m is the total number of constraints, δ_i is 1 if constraint *i* is violated and 0 otherwise, w_i is a weight factor for constraint *i*. $\Phi_i(d_i(S))$ is a function of amount of violation of constraint *i* introduced by solution S. B is a penalty threshold factor. δ_s takes the value of 1 if solution S is infeasible, otherwise 0. V(g) is an increasing function, where g indicates the current number of generation.

Annealing penalty is based on the idea of simulated annealing (Carlson and Shonkwiler, 1998). In this approach, the next generation is created using the best solution in previous generation. Therefore only single instance of a feasible individual is enough to operate annealing penalty. This penalty also increases over time and in the last generations, the infeasible individuals are heavily penalized. The objective function applied annealing penalty is presented in Equation 3.5 (Carlson and Shonkwiler, 1998).

$$fitness(\vec{x}) = \mathrm{A} \cdot f(\vec{x}) \qquad (3.5)$$

where A depends on two parameters: M, which measures the amount of violation of a constraint and T, which is a function of the running time of the algorithm. T tends to zero as evolution progress. Therefore the initial penalty factor is small and it increases over time as illustrated in Equation 3.6.

$$A = e^{-M/T} \tag{3.6}$$

The cooling schedule T can be defined as

$$T = \frac{1}{\sqrt{t}} \tag{3.7}$$

t refers to the temperature used in the previous iteration.

Adaptive penalty requires a feedback from the search process. This approach starts with a relatively small initial penalty to guarantee the adequate sampling of search space and increases or decreases it depending on the conditions (Rasheed, 1998). An objective function using adaptive penalty proposed by Smith and Tate (1993) is given in Equation 3.8.

$$fitness(\vec{x}) = f(\vec{x}) + (B_{feasible} - B_{all}) \cdot \sum_{i=1}^{n} \left(\frac{g_i(\vec{x})}{NFT(t)} \right)^k \tag{3.8}$$

where $B_{feasible}$ is the best-known objective function at generation t, B_{all} is the best (unpenalized) overall objective function, $g_i(\vec{x})$ is the amount of violation of the constraint *i*, k is a constant adjusting the severity of penalty and NFT is Near Feasibility Threshold, which is defined as the threshold distance from the feasible region (Coello, 2002).

Death penalty is the easiest way to handle constraints. In this approach, all infeasible individuals are rejected. However, this approach can not use any information in infeasible individuals. Another potential problem is that if there

is no feasible solution in initial population then the evolutionary process will stagnate.

Besides the penalization of infeasible individuals, other constraint handling techniques aim at maintaining feasible population in the evolutionary process. For this purpose, one of these techniques suggests using special representations and genetic operators. Problem specific definition of representation scheme and genetic operations will prevent the generation of infeasible individuals. It is obvious that, this approach is more reliable than the other approaches based on penalty functions (Michalewicz *et al.*, 1996). However, the generalization of these representations and operators even for similar problems is impossible.

Another constraint handling technique is to use repair algorithms. These algorithms are used to convert infeasible individuals to feasible individuals. This repaired version can either be used for evaluation only or it can also replace the original individual in the population. However, there are some drawbacks to employ repair algorithms. Such algorithms may introduce systematic bias into search and severely disturb the superior aspects of the parent solutions carried in the children, defeating the fundamental strength of the evolution (Smith and Coit, 1997). Moreover, they are problem dependent and there is no standard heuristics to design such repair algorithms. In generations, since the children often do not resemble their parents, restoring infeasible children may be as difficult as the optimization problem (Joines and Houck, 1994).

Using decoders are also used to handle constraints. In this approach, a chromosome gives instructions on how to build a feasible solution (Michalewicz *et al.*, 1996). Following these instructions always lead the evolutionary process to generate feasible individuals.

An alternative way of handling constraints is to employ multi-objective optimization techniques. The main idea behind this method is to redefine the

single objective optimization problem as a multi-objective optimization problem in which each constraint is treated as an objective. Therefore m+1 objectives are considered in optimization problem, where m is the total number of constraints (Coello, 2000).

There are many other constraint handling techniques proposed in literature. In general, all of these techniques require major or minor problem specific modifications and there is still a need for a systematic way for determining parameters of them.

3.4 MULTI-OBJECTIVE OPTIMIZATION USING EVOLUTIONARY ALGORITHMS

Genetic algorithms require scalar fitness information to operate, which means that when approaching multi-objective problems, it is needed to perform a scalarization of the objective vectors. Therefore, in order to handle multi-objective optimization problems, some techniques are suggested in literature. Coello (1996) categorized these techniques as; use of aggregating functions, non-pareto approaches and pareto based approaches.

Aggregating functions aim at combining different objective functions into a single formula. The first method in this category is "weighted sum approach". This approach requires assigning weights indicating the importance of the assigned objective. Then the objectives are combined. The difficulty in this approach is the determination of weights under absence of enough information about the problem.

"Reduction to a single objective" is another method in this category. This method assumes all objectives except one as constraints and a single value is assigned to each of these constraints. The remaining objective becomes the fitness function of evolutionary algorithm. Then, the evolutionary process is run numerous times for different values of the constraints. Thus, a trade off

surface is developed. It is obvious that this process is very time consuming and the coding is impossible for certain problems.

An additional method in this category is "goal attainment". In this method, the problem formulation allows the objectives to be under- or overachieved, enabling the decision maker to be relatively imprecise about the initial goals. The relative degree of under- or overachievement of the goals is controlled by a vector of weighting coefficients (Mathworks, 2004b). By varying weights, the important objectives can be emphasized.

An essential property of any candidate solution of a multi-objective problem is that the solution should not be dominated. The Pareto set consists of solutions that are not dominated by any other solutions. A solution *x* is said to dominate *y*, if *x* is better than or equal to *y* in all attributes, and strictly better in at least one attribute (Anderson, 2001). Figure 3.17 illustrates Pareto optimal solutions of a bi-objective (f_1, f_2) minimization problem. Each objective function is represented on a separate axis. In such a problem, since there is no point down and to the left of them in the graph, solutions 1 and 3 are Pareto optimal solutions. Solution 2 does not offer any smaller value for neither of the objectives. Therefore it is dominated by Pareto optimal solutions. Multi-objective optimization problem aims at obtaining a set of Pareto optimal solutions.

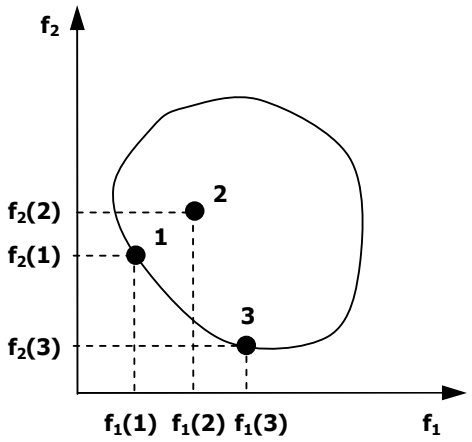


Figure 3.17 Pareto optimal solutions of a bi-objective minimization problem.

Another category in multi-objective optimization consists of non-Pareto approaches. VEGA (Vector Evaluating Genetic Algorithm) developed by Schaffer (1985) can be given as an example for these approaches. This algorithm generates a number of sub-populations at each generation. Each individual objective is assigned to one of these sub-populations as the evaluation metric. As the individuals are selected using this metric in sub-populations, in order to perform crossover and mutation operations in usual way, all sub-populations are mixed again (Anderson, 2001).

A different non-Pareto technique is lexicographic ordering. This technique ranks the objectives in the order of importance. The optimum solution is found by minimizing the objective functions, starting with the most important one and proceeding according to the order of importance of the objectives.

The last category comprises pareto based approaches. The first Pareto based approach in multi-objective optimization was proposed by Goldberg (1989a). Basically, these approaches find the individuals, which are Pareto non-dominated by the rest of the population. Then, the highest rank is assigned to these individuals and these individuals are removed from the population. Another set of Pareto non-dominated individuals are determined from the remaining of the population and the next highest rank is assigned to them. This process repeats until the whole population is ranked as presented in Figure 3.18. After that, the genetic operations are performed on this ranked population. In the literature, there are also some variations of pareto based optimization such as MOGA, NSGA and NPGA (Coello, 2002).

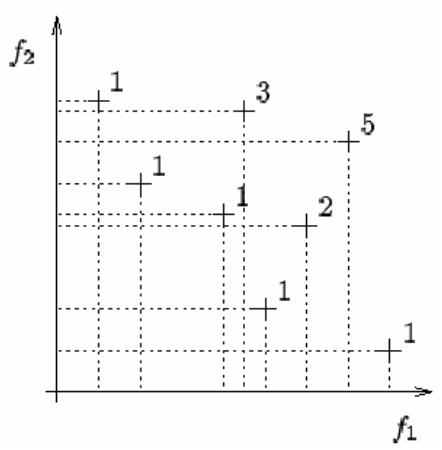


Figure 3.18 Pareto ranking (Fonseca and Fleming, 1993).

CHAPTER IV

EVOLUTIONARY DESIGN OF ENGINEERING PRODUCTS AT FUNCTIONAL LEVEL

Roston (1994) indicated the similarity between a mechanical device and a computer program in his Ph.D. thesis. While in a computer program, data is operated on by functions to produce results, in a mechanical device, physical objects are acted on by forces to produce an expedient effect as shown in Figure 4.1. Design of either of a computer program or a mechanical device seeks for a suitable mapping between the available inputs and desired output.

Figure 4.1 Black-box representations of a computer function and a mechanical device (Roston, 1994).

To the question of "How can computers learn to solve problems without being explicitly programmed?", an answer has been raised by the theory of genetic programming (GP) (Koza, 1992; 1994). This theory aims at programming without need of a human programmer. To accomplish this aim, it starts with randomly generated primitive program codes and breeds this population using the principle of survival of the fittest and two genetic operations; crossover and mutation.

Aforementioned similarity gives the idea of application of the automated programming theory in software development to product development as

automated design. There exist numerous attempts to automate design process of engineering products. Some of them have been reviewed in Chapter 2. Almost all of these attempts have focused on physical design issues such as catalog search for component selection or parameter optimization. However, there exists no previous effort to design of a functional topology of an engineering product using evolutionary algorithms. In this chapter, a theory on evolutionary design at functional level is addressed.

Before giving the details of this theory, some benefits of employing evolutionary approach for functional design issues can be summarized as follows;

- In the literature, there exists no functional reasoning technique extracting design alternatives in a solution neutral way (i.e. without referring any physical solution) except Pahl and Beitz's approach (1988). However, Pahl and Beitz's approach (1988) does not consider the law of vertical causality in the decomposition of design tasks. Moreover, this approach does not propose a heuristic to prevent the design from including overlapping functions. Finally, this approach requires one to one correspondence between each function and a physical component to generate physically realizable designs. Evolutionary approach overcomes these difficulties by working on initial populations, whose individuals are constituted by atomic functions. The atomic functions do not require hierarchical decomposition. Moreover, the independence of atomic functions and physical achievability of them are assured by a well structured functional vocabulary. In addition, the required heuristic to generate workable function structures is inherent in the structures of viable individuals in initial population.
- Generation of creative designs require the changes at functional level. Therefore, it will be more effective to apply evolutionary methods directly at functional level than at physical level and expecting alterations in functional topology.

- Functional search space consists of infinitely many solution alternatives. Evolutionary methods search for the most promising solutions incorporating the guided and random search strategies. This makes possible to extract solution suggestions difficult to find by designers, who consider alternatives limited to only those conceived by them.
- In the evolutionary methods, by means of a properly defined evaluation measure, solution alternatives are generated independently from the designer's biases and foreseeing ability.
- In order to generate optimum designs, evolutionary approaches do not require problem specific knowledge (i.e. understanding the working principles of extracted solution). Therefore, evolutionary methods search solutions in a domain independent space. This feature of evolutionary methods supports the domain independent nature of functional design process.
- Evolutionary methods carry out synthesis and analysis automatically. Therefore the designer spends his/her time on definition of a proper evaluation measure and construction of the initial population.
- Computer implementation is one of the essentials of automation. Unlike some exhaustive search strategies (e.g. breadth-first search, depth-first search etc.), evolutionary techniques are computer implementable for very large solution spaces.

In the evolutionary design process, basically the general procedure shown in Figure 3.2 is operated. Evolutionary design starts with the creation of an initial population. All individuals in this population are evaluated by using an objective function and a fitness value is assigned to each of them. By using these fitness values and the average fitness of the population, the first genetic operation, reproduction is performed. Reproduction operation determines the fitter individuals in the current population and copies these individuals to the next generation with a probability proportional to their fitness. Therefore, fitter individuals in the population survive and others become extinct.

The next step is the modification of the individuals of this new population. This step comprises two genetic operations: crossover and mutation. The crossover operation aims to generate new individuals by crossing the selected fitter individuals. Crossover operation starts with two parents and ends with two new children. As parents are chosen proportionate to their fitness values, crossover points on these parents are determined randomly. Then parent individuals are crossed at that point and two new children emerge in the new population. These operations on population are repeated until satisfying the termination criterion. This algorithm of evolutionary design process is presented in Figure 4.2.

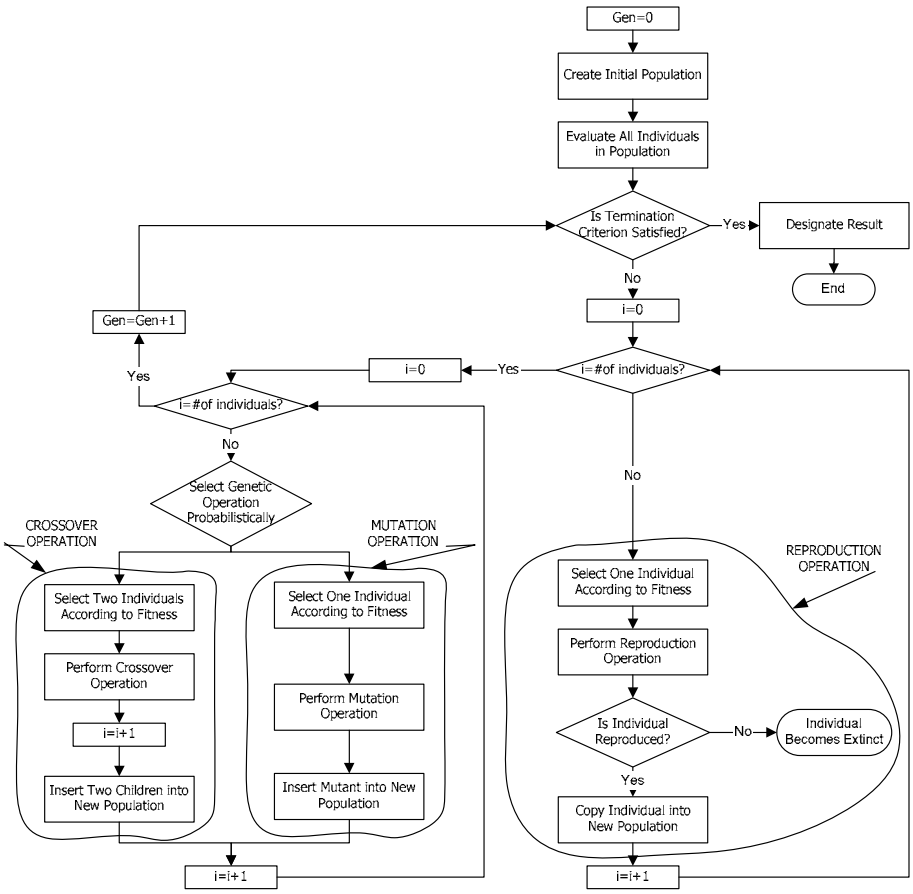


Figure 4.2 Algorithm of Evolutionary Design process.

In order to operate this algorithm, evolutionary design requires the completion of the following steps.

- A representation scheme covering all potential solutions in the search space must be developed.
- Using this representation scheme, an initial population must be constructed.
- An evaluation metric rating individuals in terms of their fitness values must be described.
- Genetic operators that alter the composition of children must be defined.
- Values for various parameters of evolutionary process such as population size, probabilities of applying genetic operators, an error value or maximum number of generations as a termination criterion etc. must be assigned.
- Finally, computer implementation of evolutionary design process is crucial for automation.

4.1 REPRESENTATION SCHEME

In evolutionary design process, representation scheme is a means to represent solutions of the design task. As in the case of other evolutionary methods, this representation scheme has to be general enough to cover all possible solutions and permit the generation of new designs. It also has to be eligible to computer implementation.

Development of a representation scheme and a formal grammar is essential for evolutionary design process. It is expected that functional formation of the generated artifact using this scheme must obey some strict grammatical rules. This requirement guarantees to obtain the syntactically correct design solutions at the end of genetic operations.

Different grammars are required for different phases of the design process from functional design to form design. In general, as the construction of the grammar is easier at abstract phases due to increasing complexity at latter stages, the performance evaluation of the generated artifacts becomes more difficult than at concrete phases (Roston, 1994). The main reason of this is the high dependency of the performance calculations to the component selections in physical domain.

In the construction of representation schemes and grammatical rules, the biological terms genotype and phenotype correspond to two different views used in evolutionary strategies.

In biology, genotype is simply the coded representation of an individual. Phenotype is the "outward, physical manifestation" of the organism such as physical parts, structures, tissues, organs and behaviors; anything that is part of the observable structure, function or behavior of a living organism (Blamire, 1997). Naturally, genotype requires decoding to produce phenotype.

In artificial evolution, genotype representation is employed by genetic algorithms. In genetic algorithms, the chromosome in the form of a fixed length character string represents the encoded artifact. However, in the genetic programming, the artifact itself (phenotype of artifact) is represented by a tree structure. As genetic algorithms are powerful especially in parameter optimization tasks, handling topology optimization problems becomes easier using genetic programming. For that reason, tree based representation scheme is used at early attempts in construction of evolutionary design process (Güroğlu and Erden, 2003). The details of this representation scheme and difficulties encountered in its application are given in following paragraphs.

Rooted tree representation requires identification of the function and the terminal sets appropriate to problem domain. As the functions become the internal nodes, the terminals are the leaf nodes in the tree structure. The

functions, being a part of the representation scheme, are different from functions constituting an artifact. As the first one configures the terminals in the representation, the second one identifies the operations in an artifact to generate the desired output.

According to the study of Stone *et al.* (1998) on functional dependencies, function chains constituting artifacts are grouped into two namely; sequential and parallel chains. In sequential function chains, the sub-functions must be carried out in a specific order to generate the desired result. Functions in a sequential chain use the output flow of previous function as its input flow. Parallel chains comprise sets of sequential chains sharing one or more common flows. By means of this classification, functions in representation scheme of evolutionary design are determined as "SEQ" and "PAR" corresponding to sequential and parallel chains respectively.

In the evolutionary design, terminals of tree based representation are the operations, which are performed by the designed artifact, to obtain the desired output flow from available input flows. The terminal set of representation scheme consists of functions and flows defined in reconciled functional basis developed by Stone and Wood (1999) and Stone *et al.* (2001). This basis provides the designer with a standard language for the description of artifacts. By using this basis, a terminal node is represented in the form of a verb-object (i.e. function-flow) pair such as "import electricity" or "guide particles". Both of the function and the flow can be selected from any of the three levels presented in Tables 2.1 and 2.2 depending on the specification desired. After the definition of the terminals, these terminals are placed to leaves of the rooted tree structure.

An illustration of tree based representation for a CD player is given in Figure 4.3. SEQ function at the root of the tree states that all sub-functions are operated in a sequential order. These sub-functions correspond to the modules of CD-Player such as supply electricity module, drive module, digital data retrieval module and sound conversion module.

In this representation scheme, in order to construct a syntactically correct configuration, all sequential and parallel modules should be placed in correct order. This order is determined by the flows on modules. This requirement violates the closure property of hierarchical representation in genetic programming theory. To correct this deficiency, an extension of genetic programming called "strongly typed genetic programming" (STGP) is employed. Strongly typed genetic programming is an enhanced version of genetic programming which enforces constraints on data types (Montana, 1995). Therefore, strongly typed genetic programming provides the hierarchical representation scheme with being discriminatory about input data types of functions to generate viable individuals.

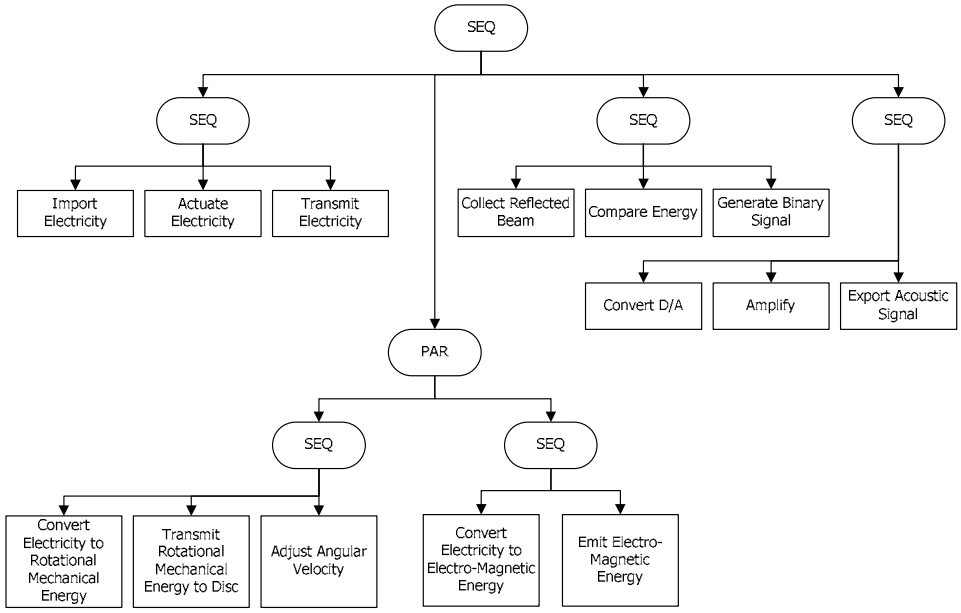


Figure 4.3 Tree representation of CD player.

Although the violation of closure property is prevented by introducing the concept of STGP, tests performed using this representation scheme indicated that there are some drawbacks of describing artifacts using functional dependencies. It was observed that the two definitions in the function set are

not enough to reveal all relations between functions in an artifact. In order to cover all relations, in addition to sequential and parallel function chains, Ulrich and Eppinger (1999) have proposed a third definition called "coupled function chains". Even though the coupled function chains are not found on the same flow, they need the completion of each other to complete their tasks. However, as this third definition assists revealing all relations between the functions, it makes the representation more complex especially for genetic operations.

Another shortcoming of tree based representation scheme originates from its structural characteristic. In tree representation, each node has a single ancestor and many descendants (i.e. children nodes). In some problems, a node may need to have more than one ancestor. Especially in functional description of artifacts, some function chains require multiple flows to operate. Then, the tree based representation becomes inappropriate for describing the relations between these function chains. In order to overcome all of these difficulties, a graph based representation scheme was proposed for the evolutionary design process.

Although graph based representation is not as common as the others, it is generally preferred in topology design and optimization problems such as truss design (Sushil, 1997), integrated circuit design, network routing (Hobbs and Rodgers, 1998), scheduling (Özdemir and Mohan, 2001) or nanotechnological molecule design problems (Lawton and Wipke, 1999).

A graph is constructed by connecting a set of vertices through edges. Each edge connects two vertices. If the edges are directed, the graph becomes a directed graph. Otherwise the graph is an undirected graph (Al-Hakim *et al.*, 2000).

In graph-based evolutionary design, solutions are represented by graph-structured individuals and all evolutionary operations are performed on these individuals. Functions and flows in an artifact correspond to the vertices and

edges in the graph respectively. The sequence of operations of functions is expressed using directed edges and cycles representing flows in the artifact. Therefore all flow motions in an artifact can be described in the functional model using this representation. In addition, the function chains requiring multiple input flows can easily be defined. In Figure 4.4, graph-based functional representation of a CD player is illustrated. In this product, the input flows are solid (CD), human energy and electricity, the desired output flow is acoustic energy. These flows are at the boundaries of the system and present the interactions between the environment and the product.

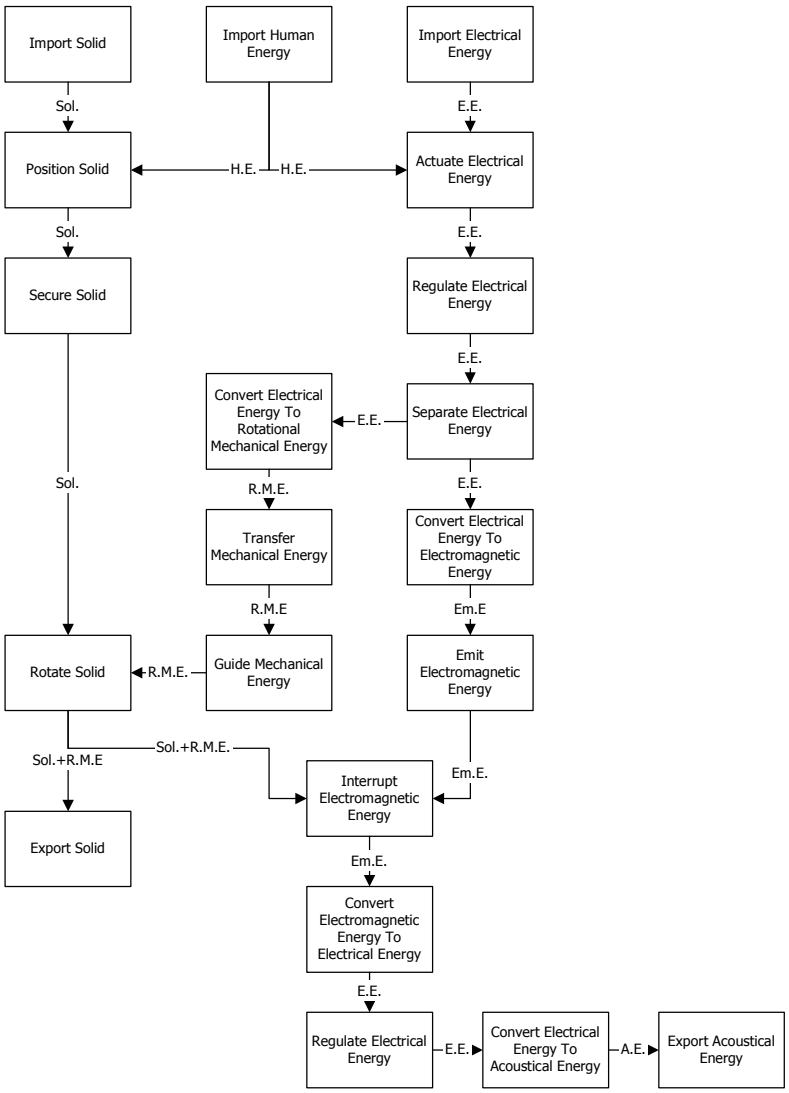


Figure 4.4 Graph based representation of a CD player.

4.2 CREATION OF AN INITIAL POPULATION

In evolutionary methods, the second step after the determination of representation scheme is the creation of an initial population. Initial population can be generated randomly or a consciously prepared population can be adopted. Although using randomly generated initial populations is very common in applications of evolutionary methods, evolutionary design prefers to work on a predetermined population.

Individuals consisting of function clusters must obey precise grammatical rules to represent a workable solution alternative. Certain functions in the reconciled functional basis, which is the vocabulary of formal representation scheme, are limited to operate on certain types of flows. In evolutionary design, this can be provided with either operating on already working function structures or operating on a randomly created initial population by the help of restrictions on genetic operations and powerful constraint handling algorithms. As a randomly created population requires the definition of many constraints and decreases the efficiency of evolutionary process, working function structures inherently satisfy these constraints. The way of defining some heuristics or restrictions to construct working individuals makes the design process unmanageable.

Moreover, random generation of individuals may result in lack of diversity in the population (i.e. not having a single feasible individual). Lack of diversity in a population causes all individuals to have similar or equal fitness values. Therefore it seriously limits the applicability of evolutionary process in highly constrained search spaces.

However, some difficulties also appear in the use of designer defined initial populations. First of all, an initial population should include all function chains and flows required to construct at least one solution alternative for the design task. For instance, if an initial population completely consists of pneumatic systems, it can not be expected to produce a solution requiring an electro-

optic system (e.g. laser range finders, optical sensors etc.) from this population. In order to overcome this problem, it would be beneficial to select a part of the individuals among the members of similar product families to the goal product. Intuitively, the proper selection of the remaining individuals from other product families will provide the design process with creativity.

The studies performed on product design based on modular architecture indicate that products in the same product family handle many common modules to create product variants. In the study of Chandrasekaran and Stone, (2001), these modules are classified into three major categories: operational modules, carrier modules and form defining modules. Operational modules are responsible for importation, conversion, transmission and end-utilization of the energy. In other words, these modules run the product. The carrier modules are responsible for carrying energy flow in a product. Finally, form defining modules influence how the product looks physically. In Table 4.1, some consumer products are examined based on the presence or absence of certain operational modules, which are the main modules in the run of the product.

Table 4.1 Product-Operational Module Matrix adopted from Chandrasekaran and Stone (2001).

Operational Modules / Products	Temperature Control	Thermal	Electrical	Rotation	Rotation Control	Magnetic
Electric Wok	X	X	X			
Toaster	X	X	X			X
Coffee Maker	X	X	X			
Kettle	X	X	X			
Iced Tea Maker		X	X			
Café Trio	X	X	X			
Pop Corn Popper	X	X	X	X		
Blender			X	X	X	
Juicer			X	X		
Dough Machine			X	X	X	

This table proves that, from operational point of view, many modules are repeated in several products belonging to similar product families. This emphasizes the need for similar individuals to the goal product in construction of initial population.

Another difficulty emerging in the selection of individuals probably stems from the designer's bias. Creativity as an evaluation measure, which is detailed in Section 4.3.1.2, highly depends on the probability of occurrences of functions and flows in initial population. Therefore the designer should perform the individual selection process objectively to facilitate the diversity in initial population.

The number of individuals in the initial population and the size of the individuals are other criteria, which should be considered by the designer. In order to generate high quality solution alternatives by effectively using computational resources, the number of individuals in the initial population and the size of the individuals should be properly determined by the designer.

4.3 EVALUATION METRICS

In order to achieve automatic evaluation of the generated design alternatives, an evaluation metric, which will be used as an objective function, is required. This objective function must express the designer's intent and have the ability to guide evolution process to generate feasible alternatives. It should also be capable of evaluating all possible individuals that can be encountered in the steps of evolution.

Although evolutionary design process can be operated using a single objective function, the adopted evaluation measure should be suitable to handle multiple objectives to search optimum alternatives for multi-criteria design tasks. Moreover, constraint handling, one of major concerns of designers, should also be considered in the development of this measure.

4.3.1 Objective Functions in Evolutionary Design

Cost and performance of an artifact are two of the most common evaluation criteria in design problems. However, in early phases of the design process, both of these criteria are difficult to observe. Although, some function costing methodologies have been proposed to make a rough estimation of the product cost in the literature, all of these methodologies require physical embodiment of the product to operate. In the same way, performance calculations strictly depend on the means selection for intended functions. Evolutionary design process at the functional level requires evaluation measures independent of physical solution alternatives. Therefore, "complexity" and "creativity" as two means-independent evaluation measures are proposed in this study.

Complexity as a measure aims at guiding the evolutionary process to generate simple designs in size based on the assumption that simple solutions have a greater probability of success. This measure focuses on the complexity of the artifact rather than the amount of effort required to design or manufacture it.

The other measure, creativity is handled as "novel combinations of old ideas" as considered in Boden's studies (1991). In evolutionary conceptual design, these combinations take place at functional level. This measure leads the evolution process to generate original and useful design alternatives.

4.3.1.1 Measuring the Complexity of an Artifact

In order to measure the complexity of an artifact, an information-based metric is employed. According to this metric, structural complexity of a design is interpreted as a function of its information content (Summers and Shah, 2003). This approach allows comparing generated design alternatives according to their complexity level.

In the study of Summers and Shah (2003), a measure of structural complexity for hierarchically represented systems has been proposed. This measure is

based on the information theory (Klir and Folger, 1988). The complexity measure employed in the evolutionary design process is mainly inspired by this structural complexity definition (Güroğlu and Erden, 2004). The changes in the formulation are due to the differences in the representation methods. As Summers and Shah (2003) handle a design hierarchically, in the evolutionary design, functional structure of an artifact is described at a single abstraction level. The abstraction level in a representation is guaranteed by using a standard vocabulary.

The information included in a design appears in two different forms: operands (entities) and operators (relationships). In representation scheme of evolutionary design, as operands correspond to functions, operators represent flows. A design artifact is a sequence of instances of operands and operators. The information content of this sequence is calculated by joint entropy equation.

The size of vocabulary "λ" is the summation of the number of unique operands "ρ" and the number of unique operators "ν".

$$\lambda = \rho + \upsilon \tag{4.1}$$

Assuming that all operators and operands are independent and identically distributed, the probability of occurrence of each variable is calculated as;

$$P(x) = \frac{1}{\lambda} \tag{4.2}$$

Information content of a specific sequence of operands and operators is calculated through joint entropy equation;

$$H = -\sum \left(P(x) \cdot \ln\left(P(x)\right) \right) \tag{4.3}$$

According to this equation, the entropy of an artifact indicates the complexity of design and it decreases as more information is collected (i.e. as the probabilities involved increase). This measure is completely independent of the form of the artifact. It basically focuses on the degree of interconnections and size.

4.3.1.2 Measuring the Creativity in an Artifact

Creativity is often associated with unpredictability. According to Boden (1991), creative ideas are brought into being by unusual and surprising combinations of ideas. By definition, creativity is identified due to the improbability of combination, which brings novelty.

However, every surprising or unpredictable idea cannot be identified as creative at the same time. A creative idea must be useful, illuminating, or challenging in some way (Boden, 1991). Therefore, constraints are essential for identification of creativity in evolutionary design process. Constraints act as the criteria of judgement. Random process without constraints, which give birth to unpredictable and interesting ideas, causes first time curiosities. However, a process, which is carried out with respect to constraints, can give novel and useful ideas.

Genetic operations (i.e. mutation and crossover), which are the essence of evolutionary design process, have random characters. For instance, mutations do not happen for their capacity to bring individuals who can survive. While mutations randomly generate variety in chromosomes, natural selection lets the fit individuals to survive. Natural selection provides the constraints necessary for fit individuals. On the other hand, crossover process operates in a more constrained nature since it is carried out between two individuals, which have high survival value. However, even in crossover operation, the chromosomes are broken from random points.

"Improbability of combination", which is mentioned in Boden's study (1991), can be handled as "low probability of occurrence" in evolutionary design process (Güroğlu *et al.*, 2005). The individuals, which have low probability of occurrence, are regarded as the ones, which have original character with respect to the individuals included in initial population.

Since the design process is based on identifying necessary functions and flows and building a plausible structure from them, usefulness of the built structure appears as the ultimate outcome. Designer's goals and constraints determine the usefulness of generated solution.

Creativity in generated artifacts is a relative measure. It can be computed by comparing the generated individuals relative to the individuals in the initial population. Therefore it is formulated considering the instances of unique operands and operators in the initial population.

The total number of instances (N) of the whole elements in the vocabulary is the summation of instances of each operands and operators in the initial population.

$$N = \sum_{i=1}^{n} \left(\sum_{j=1}^{m} \Phi_i \left(F_j \right) + \sum_{k=1}^{l} \Psi_i \left(O_k \right) \right) \tag{4.4}$$

where n, m and l correspond to number of individuals in the initial population, number of unique operands and number of unique operators in the vocabulary respectively. Φ and ψ indicate number of instances of each unique operand ($F_{j=1..m}$) and operator ($O_{k=1..l}$) in the i^{th} individual.

It is assumed that all operators and operands are independent. Unlike the complexity measure, they are not identically distributed. Therefore, the probability of occurrence of each operand and operator in the initial population are calculated as indicated in Equations 4.5 and 4.6 respectively.

$$P(F_j) = \frac{\sum_{i=1}^{n} \Phi_i(F_j)}{N} \tag{4.5}$$

$$P(O_k) = \frac{\sum_{i=1}^{n} \Psi_i(O_k)}{N} \tag{4.6}$$

Then for a specific sequence;

$$H = \sum \left(\ln(P(F)) + \ln(P(O)) \right) \tag{4.7}$$

This measure provides the designer with a rough idea about the originality involved by generated artifact comparatively to the individuals in initial population. However, in order to assess this value as a measure of creativity, the generated individual should satisfy the designer's goals and constraints.

However, stand-alone application of the creativity measure may cause the possibility of occurring unpredictability steming from complication (Boden, 1991). Non-deterministic nature of evolutionary process can be misleaded to generate complicated solutions to reach creative ideas. Therefore the complexity measure mentioned in Section 4.3.1.1 is required to control the level of complication of generated artifacts.

In order to make the problem manageable, in calculation of the complexity and the creativity measures, it is assumed that operands and operators are mutually exclusive (usually not the case in design). As it is mentioned by Summers and Shah (2003), this assumption make these measures independent of domain restrictions and heuristics which require use of conditional probabilities. Since all operands and operators are handled as being mutually exclusive, the calculations result in relative measures. These measures are also appropriate for evolutionary process.

4.3.2 Handling Constraints in Evolutionary Design

Constraints may arise in many forms in design. They determine feasible and infeasible solution alternatives for the design task. Different constraints can be defined depending on the designer's needs.

In the evolutionary design, the design task itself is treated as a constraint, which has a higher priority over the others. At functional level, the design task is simply defined by a black box using available inputs to give desired outputs. The design task as a constraint necessitates that all feasible solution alternatives should import these available input flows and export the expected output flows. This constraint forces the evolution process to generate feasible solutions by penalizing the infeasible individuals. A penalty value is added to the fitness values of the infeasible individuals by considering the amount of constraint violation. The amount of constraint violation is determined by the number of missing flows at the black box representation of the generated artifact. Penalties make feasible individuals fitter than the others. For instance, if the goal product is a kettle, the individuals, which take liquid and electricity as the inputs and give hot liquid as the output are feasible. All of the other individuals are infeasible and penalized. Each expected flow both at input and output of the product is accepted as a distinct requirement, which must be satisfied by the goal product.

4.3.3 Handling Multiple Objectives in Evolutionary Design

The easiest and perhaps the most widely used method to handle multi-criteria design problems is the weighted sum approach. The logarithmic function in both of the evaluation criteria mentioned above makes them additive. Therefore, evolutionary design method adopts this approach.

The weighted sum method involves the aggregation of all the objective functions using different weighting coefficients for each of them. Therefore

the multi-objective optimization problem is transformed into a scalar optimization problem (Coello, 1996):

$$\min \sum_{i=1}^{k} \omega_i \cdot f_i\left(\bar{x}\right) \tag{4.8}$$

where $\omega_i \geq 0$ are the weighting coefficients representing the relative importance of the objectives. Therefore this method provides the designer with a way for earlier articulation of his/her preferences (Anderson, 2001). In general

$$\sum_{i=1}^{k} \omega_i = 1 \tag{4.9}$$

The results are highly dependent on the weighting coefficients. Therefore in order to assign weighting coefficients, it is required to solve the same design problem for many different values of ω_i. Moreover, since objective functions have generally different magnitudes, in order to apply this method, they must be normalized at first.

4.4 FORMULATION OF OPTIMIZATION PROBLEM

The constraints and objective function are combined to give the following optimization problem:

minimize

$$F\left(\overline{X}, \rho_k\right) = f\left(\overline{X}\right) + G\left(\overline{X}\right) \tag{4.10}$$

subject to

$$O_I \subseteq \left(O_A \cup O_D\right)$$

$f(\overline{X})$: fitness value of the evaluated individual, $\overline{X}$. This value is calculated by using Equation 4.11.

$$f\left(\overline{X}\right) = \omega_1 \cdot \sum_{1}^{n_{total}} \left(\frac{1}{\lambda} \cdot \ln\left(\frac{1}{\lambda}\right)\right) + \omega_2 \cdot \left(\sum_{i=1}^{n_{functions}} \left(\ln\left(P\left(F_i\right)\right)\right) + \sum_{j=1}^{n_{flows}} \left(\ln\left(P\left(O_j\right)\right)\right)\right) \qquad (4.11)$$

The variables are defined as follows;

- w_k: weight of k^{th} criterion in objective function
- $n_{functions}$: number of functions in evaluated individual, $\overline{X}$.
- n_{flows}: number of flows in evaluated individual, $\overline{X}$.
- $n_{total}=n_{functions}+n_{flows}$
- λ: size of the vocabulary
- $P(F_i)$: probability of occurrence of i^{th} function (operand) in the initial population
- $P(O_j)$: probability of occurrence of j^{th} flow (operator) in the initial population
- O_I: flows of evaluated individual, $\overline{X}$.
- O_A: available flows
- O_D: desired flows
- $G(\overline{X})$: static penalty function

Optimization function can be expanded by adding new criteria and constraints depending on the design task.

4.4.1 Penalty Function

In the evolutionary design, the static penalty function is employed. Thus a constant penalty is applied to the fitness values of infeasible individuals without considering the current generation number.

As it is discussed in Section 4.3.2, all feasible individuals must have the same input and output flows as the goal product. This necessity is handled as two separate constraints in the evolutionary design process.

The penalty function for these two constraints is calculated as:

$$G\left(\overline{X}\right) = \sum_{i=1}^{2} C_p \cdot \delta_i \tag{4.12}$$

where
$$\begin{cases} \delta_1 = \dfrac{F_{in\_goal} - F_{in\_generated}}{F_{in\_goal}} \\ \delta_2 = \dfrac{F_{out\_goal} - F_{out\_generated}}{F_{out\_goal}} \end{cases}$$

- $G(\overline{X})$: penalty value of the evaluated individual, $\overline{X}$.
- C_p: the penalty constant.
- δ_i : the amount of violation of the i^{th} constraint.
- $F_{in\_generated}$: number of the same input flows in the generated individual with the goal product.
- F_{in_goal}: number of input flows in the goal product.
- $F_{out\_generated}$: number of the same output flows in the generated individual with the goal product.
- F_{out_goal}: number of output flows in the goal product.

Although, a stepwise increase in the penalty (i.e dynamic penalty) puts more selective pressure on the evolutionary process to find a feasible solution (Joines and Houck, 1994), the performed tests have indicated that static penalty is also able to cope with the probable diversity problem in an initial population. In the test runs using static penalty, it is observed that even if there is a single feasible individual in the initial population, the best individuals of all runs are feasible. However, when additional constraints are imposed on the design problem, the use of dynamic or adaptive penalties might be necessary to improve the efficiency of evolutionary design process.

4.5 GENETIC OPERATIONS

The primary operations of evolutionary strategies, reproduction and crossover are also adopted by evolutionary design method. The main idea behind the genetic operations is the same with the genetic algorithms and the genetic programming. However, there exist some changes in application depending on the graph-based representation scheme. In the following sections, reproduction, crossover and mutation operations are explained in detail.

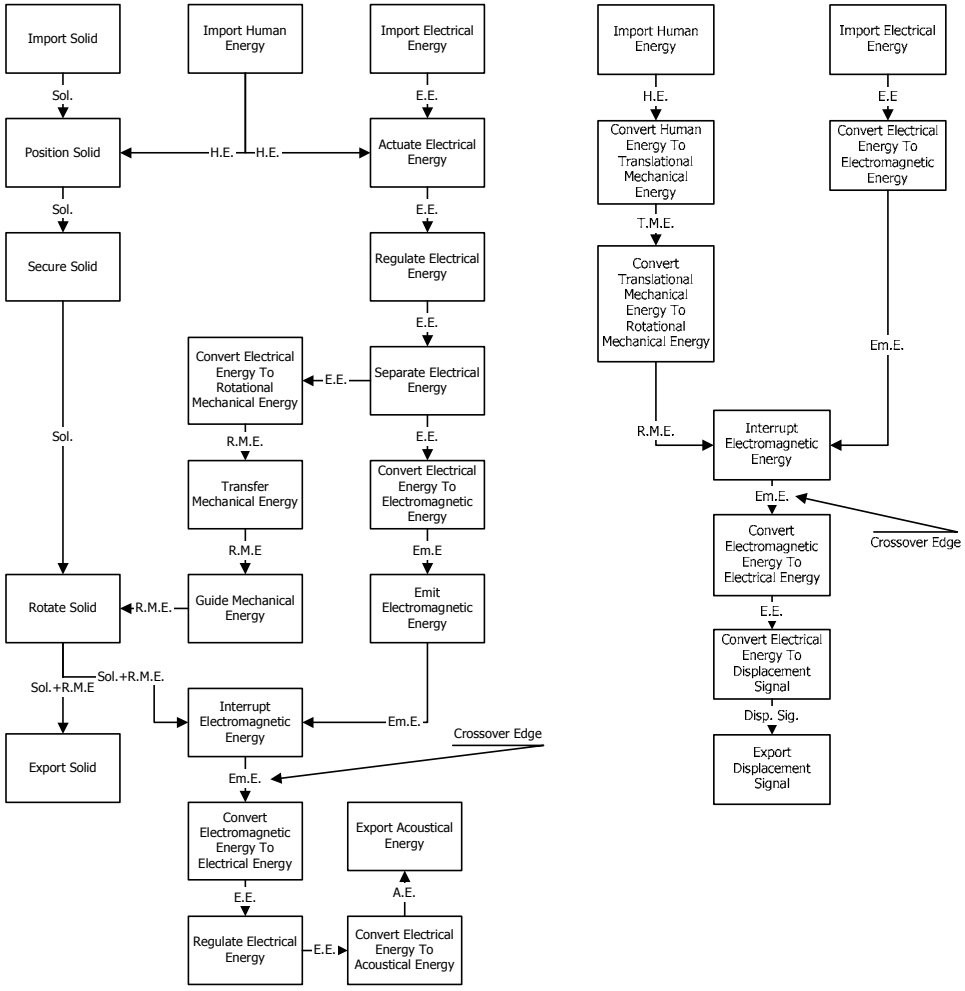
4.5.1 Reproduction Operation

After the initialization of the population, the next step is the evaluation of the individuals. Regarding the results of this evaluation, a reproduction algorithm is operated. Any reproduction algorithm such as weighted roulette wheel selection, rank selection or tournament selection etc. can be chosen by the designer. At the end of this operation, the surviving and extinct individuals are determined. Reproduction prepares the population to crossover operation. This procedure repeats itself in each generation until reaching termination criterion.

4.5.2 Crossover Operation

In evolutionary design process, a single point crossover operation is employed. This operation separates individuals into two fragments from a randomly selected crossover edge. In order to perform crossover operation, edges cut (i.e. crossover edges) should represent the same flow at both of the parents. Otherwise separated segments can not be connected in a symmetric manner. By the replacement of the separated segments, two new individuals are generated. Since the evolutionary design uses directed graphs, convenience of the direction of the edges between the mating parts is also important. Number of individuals participating in the crossover operation is determined by the crossover probability defined by the designer.

In Figure 4.5, functional structures of parent individuals participating in a sample crossover operation, CD player and Opto-mechanical mouse, are presented. It is assumed that these parents are selected based on their fitness values. Crossover points are determined by random selection between similar flows available in both of the parents. In this example, one of the electromagnetic energy flows is selected as the crossover edge for each parent. The parents are cut at these edges and each graph is separated into two. By considering the direction of the edges cut, the separated fragments are replaced.



Parent I CD Player Parent II Opto-Mechanical Mouse

Figure 4.5 Parents participating in crossover operation, before crossover.

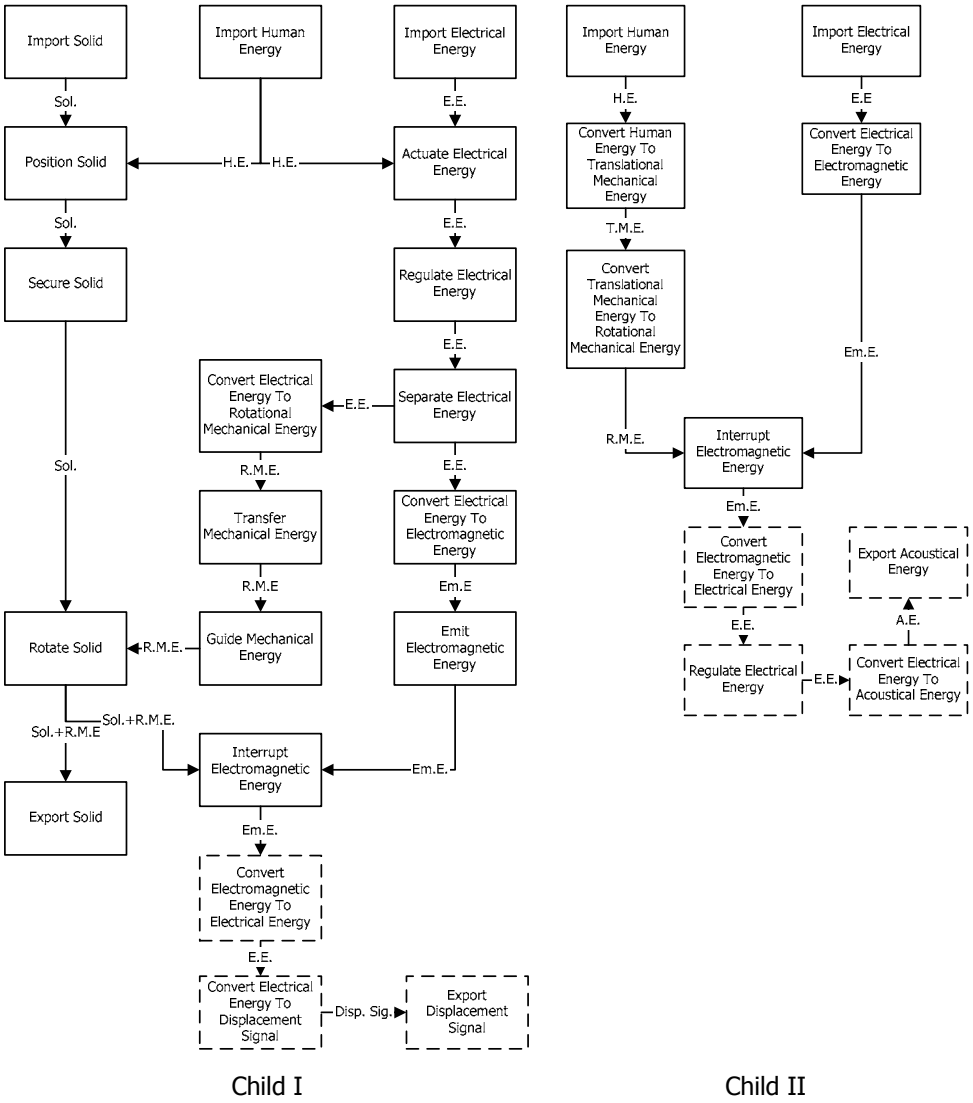


Figure 4.6 Children obtained at the end of crossover operation.

Figure 4.6 presents the individuals obtained at the end of crossover operation. The first child obtained at the end of the crossover operation can be embodied by an encoder coupled to an electric motor that is frequently used where speed control is required such as paper feeding mechanisms in inkjet printers or differential drive systems of robots. The second child looks like a musical instrument. It gives the user instant musical feedback as the user moves the mouse. At present, a similar product called music mouse for Macintosh users is commercially available in the market. Although in this example, crossover operation produced two devices serving to certain purposes, this operation

often generates conversion and transmission modules for energy, matter and information flows, which are not dedicated to a specific purpose.

4.5.3 Mutation Operation

Mutation is a secondary genetic operation. Although it is not a dominant force in evolution, it aims at restoring lost diversity in early generations of population.

In graph-based evolutionary algorithms, mutation operation replaces a graph segment bordered by randomly selected mutation points with a randomly generated graph segment. The same difficulties in random generation of the initial individuals are also valid for random generation of graph segments. Generation of operational segments (i.e. syntactically correct graph divisions) requires the definition of large amount of constraints to guide generation procedure. Otherwise mutation would require a costly repair operation. Therefore mutation operation is not adopted at the present stage of the evolutionary design process. It is operated by a crossover-driven approach.

4.6 CONTROL PARAMETERS

Control parameters adjusted by the designer determine the efficiency of evolutionary process and quality of generated designs. These parameters are described below:

- Population size: corresponds to number of individuals in the population. As the large populations improve the quality of the result, it also increases the computational cost.
- Number of generations: is the number of test, select and reproduce cycles in evolution (Roston, 1994). It is determined empirically.
- Probability of crossover: determines the number of individuals participating in the crossover operation. This value is also set empirically.

- Maximum size of individuals: basically states the maximum number of functions and flows that can be included by an artifact. It is adjusted by the designer considering the computational costs in the evolution process.

CHAPTER V

COMPUTER IMPLEMENTATION

5.1 SOFTWARE ARCHITECTURE

The software implementation of the evolutionary design theory is done at MS Visual C++ programming environment using LEDA library developed in the Max-Planck Institute of Computer Science and GALib developed at the MIT CAD Lab.

GaLib is a C++ library of genetic algorithm objects (Wall, 1996). The library supports all genetic operations and many representation types such as tree, various strings and arrays except graphs.

Another C++ library, LEDA provides efficient data types and algorithms in the fields of geometric computing, combinatorial optimization, graph and network problems (Max-Planck Institute, 2004).

In computer implementation of evolutionary design process, genetic operations defined in GALib are operated on graph based individuals represented using LEDA library in C++ programming environment. Crossover operation defined in GALib was redefined to make it appropriate for graph-based individuals. Structure of the software is presented in Figure 5.1.

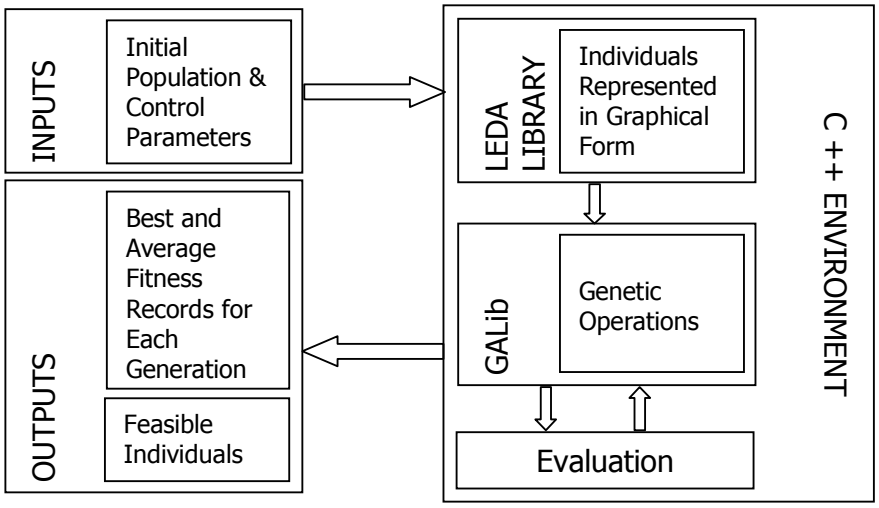


Figure 5.1 Structure of implemented software.

Using LEDA library, the graphs of the individuals that contain many different types of nodes (i.e. functions) and edges (i.e. flows) are identified by means of a structure shown in Figure 5.2. Besides connectivity map, this structure holds the types of included functions and flows by graph.

```
struct GraphStruct {
                graph *G;
                node *Nodes;
                edge *Edges;
                int *NodeInfo;
                int *EdgeInfo;
};
```

Figure 5.2 Graph representation using LEDA library.

Steady-state genetic algorithm defined in GALib is adopted in the software implementation. Instead of replacing all parents by their children as in conventional GA, steady-state GA involves keeping a specified percentage of the population and renewing the rest with the newly formed individuals. It is empirically observed that steady-state GA prevents premature convergence of population and reaches an optimal solution with fewer number of fitness evaluations (Davis, 1991).

The developed software has a modular architecture as shown in Figure 5.3. This feature makes the software eligible for future improvements especially for the definition of new objective functions or genetic operations such as multi-point crossover or mutation. New design criteria depending on the design task can be easily implemented by only an addition of a new objective function to evaluation module of the software. Moreover, the software allows changing selection algorithms used in reproduction operation by providing the designer with predefined selection schemes such as rank selector, roulette wheel selector, tournament selector or deterministic sampling selector. The control parameters (e.g. number of generations, probability of crossover etc.) of evolutionary process can also be altered by designer easily. Fitness value of the best individual and average fitness value of the population is recorded for each generation of the evolution process. All feasible individuals encountered in these generations are also presented as an output of the software.

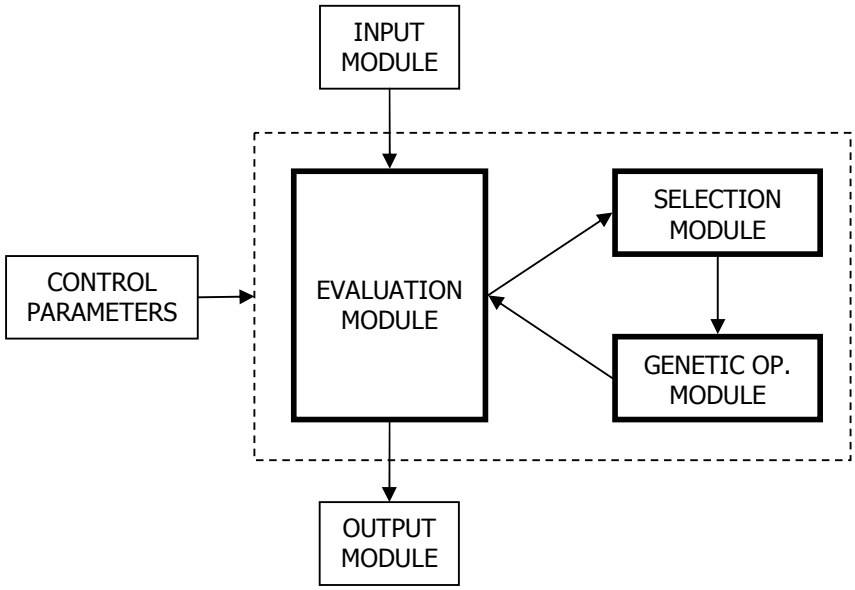


Figure 5.3 Modules of the software.

5.2 INITIAL POPULATION

The initial population is constructed by considering the criteria mentioned in Section 4.2. 100 products have been selected and functional models of these products have been built. These products are mainly consumer oriented, household appliances, hand held construction tools, mechanical or electro-mechanical devices and energy conversion modules.

The initial population comprises;

- A group of individuals selected from kitchen appliances. It belongs to the same product family of the goal product, which is designed in Section 5.4.
- Some products belonging to other product families. In the selection of these products, products consisting of similar modules are avoided. Products including distinct modules are modeled, e.g. Peltier cooler, crystal microphone etc.
- Some simple components and energy conversion modules (e.g. solenoid, photo-voltaic cell, piezoelectric actuator etc.).
- Some gauges and transducers (e.g. thermocouple, pressure gauge etc.) to increase the variety in the initial population.

5.3 EVALUATION OF INITIAL POPULATION

The initial population is evaluated with respect to both the complexity and the creativity measures proposed in Sections 4.3.1.1 and 4.3.1.2. The resulting histograms for the normalized values of these measures are presented in Figures 5.4 and 5.5 respectively.

The complexity values of the artifacts in the initial population presents a roughly Gaussian distribution as shown in Figures 5.4. This validates the unbiased construction of the initial population with respect to this criterion.

The only exception is observed in the distribution of the complexity values at the left most part. This is originated from inclusion of simple components, energy conversion modules, gauges and transducers to initial population.

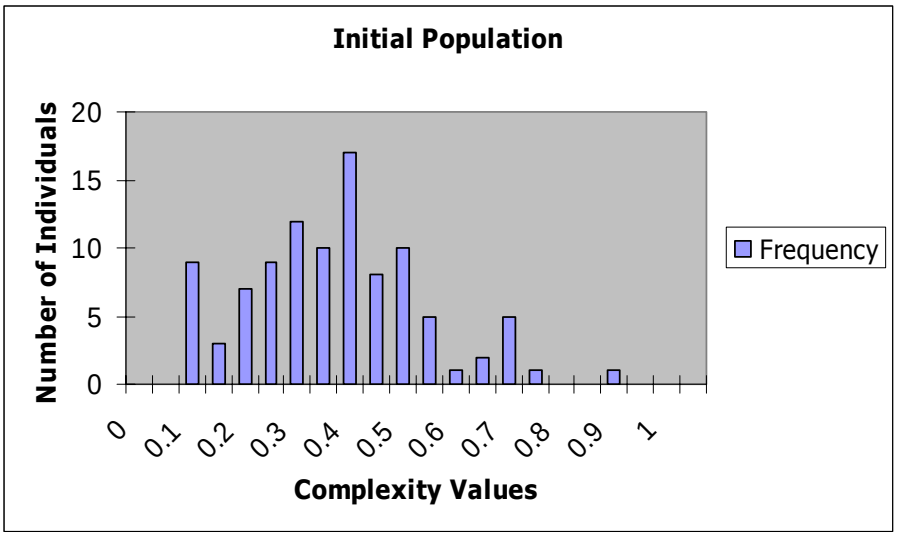


Figure 5.4 Frequency distribution of the individuals in the initial population with respect to their complexity values.

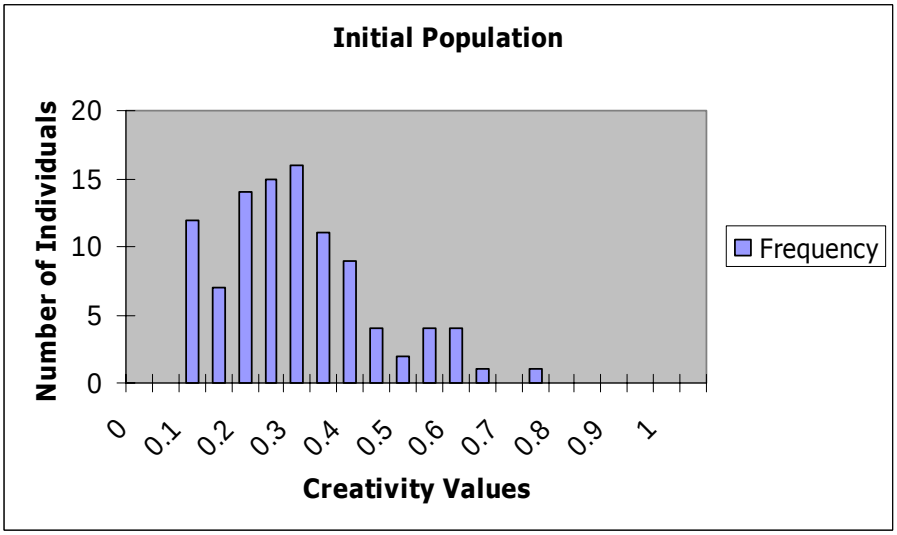


Figure 5.5 Frequency distribution of the individuals in the initial population with respect to their creativity values.

According to the creativity criterion, the individuals with a low probability of occurrence have high creativity values. There are a small number of these individuals in the initial population as shown in Figure 5.5. However, the simple individuals and the individuals including the modules, which are repetitive in the initial population, have high probability of occurrence and they constitute the significant part of the population.

In order to demonstrate the application of evaluation measures, computation results for a representative sample having 15 of the 100 products are presented in the following sections. These products consists of bath scale, magnetic door bell, hand vacuum, humidifier, electric tooth brush, electric wok, hair drier, kettle, rice cooker, electric knife, blender, deep fryer, espresso machine, coffee maker and iced tea maker.

5.3.1 Computation of the Complexity Values for the Sample Population

The complexity values of the sample population are computed and presented in Table 5.1. The values are sorted in the ascending order. The simplest design has the value of 0.217 and the most complex design has the value of 0.883.

Table 5.1 The sample population sorted with respect to the complexity evaluation.

#	INDIVIDUAL	COMPLEXITY
1	Bath Scale	0.217
2	Magnetic Door Bell	0.217
3	Hand Vacuum	0.317
4	Humidifier	0.350
5	Elec. Tooth Brush	0.350
6	Electric Wok	0.367
7	Hair Drier	0.367
8	Kettle	0.367
9	Rice Cooker	0.383
10	Electric Knife	0.433
11	Blender	0.483
12	Deep Fryer	0.517
13	Espresso Machine	0.617
14	Coffee Maker	0.650
15	Iced Tea Maker	0.883

In interpreting the results, it should be considered that the complexity measure evaluates the size and degree of interconnections in functional representation. This should not be confused with the form complexity. As suggested by Suh (1990), in the functional design, employing independent functions is essential to increase the probability of success of an artifact. In other words, independent functions decrease the difficulty in achieving the design task. On the contrary, in physical design, physical coupling is encouraged. Integration of more than one function in a single part, as long as the functions remain independent, reduces form complexity of the artifact. Therefore the complexity values presented below should not be handled through conceiving of forms of the artifacts.

The table indicates that bath scale shown in Figure 5.6 and iced tea maker shown in Figure 5.7 are the simplest and the most complex products respectively. This is an expected result when the number of functions and flows of the two products are considered.

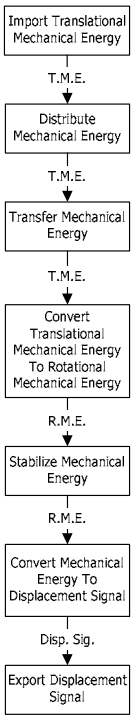


Figure 5.6 Functional structure of the bath scale.

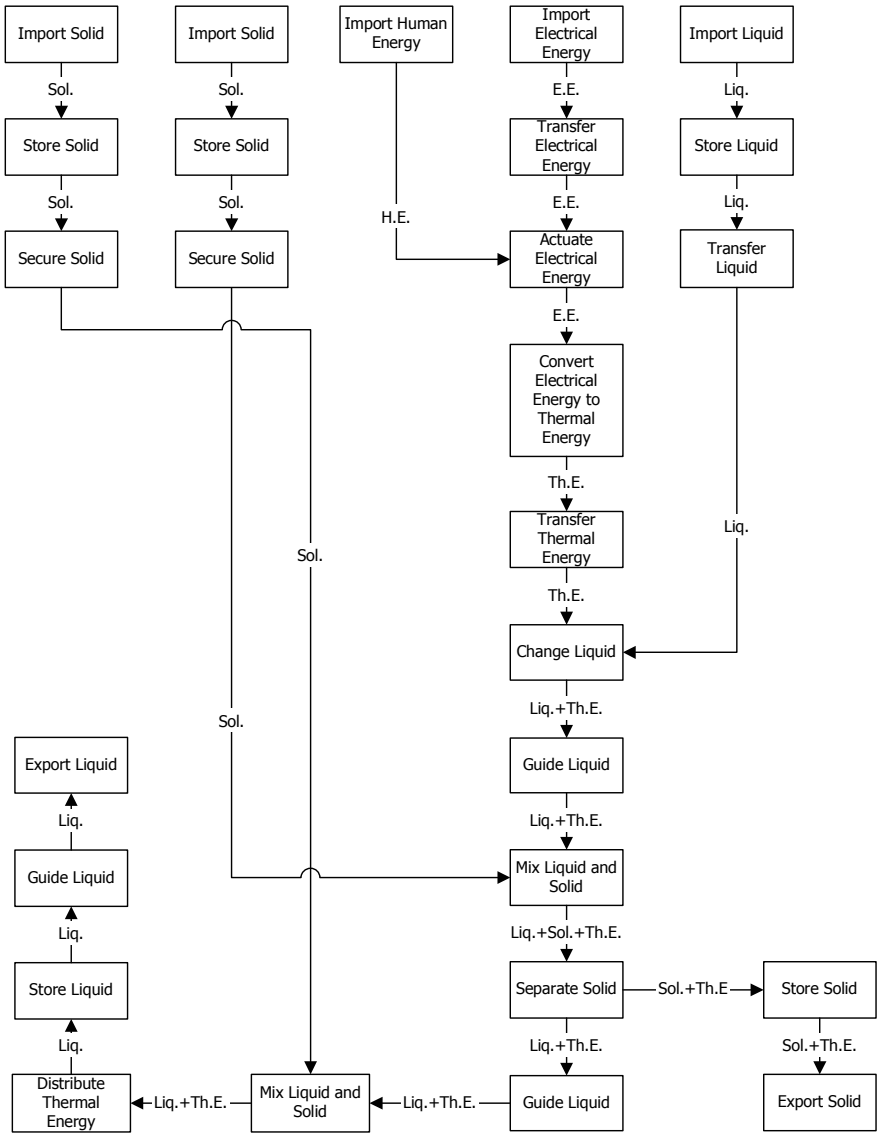
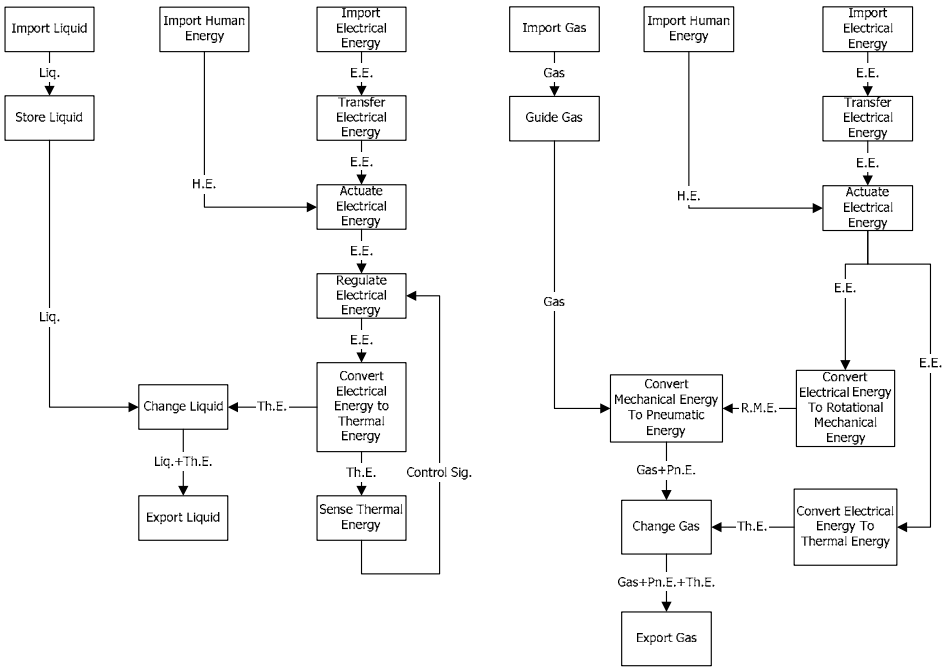


Figure 5.7 Functional structure of the iced tea maker.

Table 5.1 also illustrates that the complexities of the kettle shown in Figure 5.8a and the hair drier shown in Figure 5.8b are identical as both of these devices have the same number of functions and flows.



a. Kettle. b. Hair drier.

Figure 5.8 Functional structures of the kettle and the hair drier.

5.3.2 Computation of the Creativity Values for the Sample Population

The creativity values of the sample population are computed and presented in ascending order in Table 5.2. This measure expresses unpredictability of the designs based on the frequency of occurrence of the functions and the flows in the initial population.

The iced tea maker is now the most creative design in the population. However, it might not be the real situation. The stand alone application of creativity misleads the evolutionary process to generate designs with low probability of occurrence due to their complicated structures rather than their exceptional attributes. Therefore, for the evolutionary design process, it is a necessity to apply both of these measures together through a multi-objective optimization strategy.

Table 5.2 The sample population sorted with respect to the creativity evaluation.

#	INDIVIDUAL	CREATIVITY
1	Magnetic Door Bell	0.159
2	Bath Scale	0.199
3	Elec. Tooth Brush	0.246
4	Hand Vacuum	0.251
5	Humidifier	0.256
6	Hair Drier	0.269
7	Electric Wok	0.278
8	Kettle	0.282
9	Electric Knife	0.306
10	Rice Cooker	0.315
11	Blender	0.361
12	Deep Fryer	0.414
13	Espresso Machine	0.537
14	Coffee Maker	0.542
15	Iced Tea Maker	0.750

5.4 TEST CASE

5.4.1 Goal

As a case study, design of a product, which cooks food (e.g. potato, mushroom etc.), has been performed by using the developed software with the initial population mentioned above. The black box representation of this design task is given in Figure 5.9.

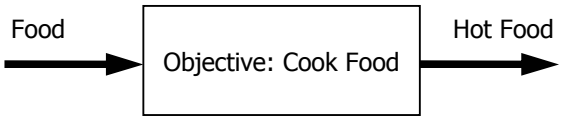


Figure 5.9 The black box representation of the design task.

The black box representation of the product presents a high level description of the design task, which consists of the necessary flows at the boundaries of the product and the objectives. In the design of a cooker, the necessary flows are food and hot food. During the evolutionary process, these flows are treated as the constraints of the design task. In order for an individual to be

feasible, it should have these flows at its boundaries (i.e. its inputs and outputs). The individuals that violate these constraints are penalized by using the penalty function proposed in Section 4.4.1.

During the tests, the available flow is simply identified as "*solid*", which corresponds to the raw food and the desired output flow is "*thermal energy* + *solid*", which represents the cooked food. It is expected that evolutionary process searches for different ways of cooking such as heating, boiling, frying baking, steaming etc. starting from the initial population. The feasible individuals present in the initial population are only rice cooker, microwave oven and electric wok.

The last step is the identification of control parameters. After performing numerous test runs, the control parameters of the evolutionary process were set to the values as shown in Table 5.3. The replacement percentage mentioned in the fourth row specifies the percentage of the population to replace at each generation.

Table 5.3 The run-time control parameters of the evolutionary process.

Size of the Initial Population	100
Maximum Number of Generations	120
Crossover Probability (p_c)	0.95
Replacement Percentage	0.10
Penalty Constant (C_p)	0.50
Termination Criterion	maximum number of generations

5.4.2 Results

In order to employ both of the evaluation measures, the weighting coefficients (i.e. ω_1 and ω_2 in Equation 4.11) of the multi-criteria objective function must be determined. Since the results of the evolution process can vary significantly as the weighting coefficients change, the necessary approach is to solve the same problem for many different values of the weighting coefficients (ω_i) (Coello, 2001). For this purpose, some test runs were again performed. The results of these test runs indicate that the evaluation measure, which has a

greater weighting coefficient than 0.5, becomes dominant and the evolution process searches for the global minimum of this measure. As a result, both of the weighting coefficients were set to 0.5. The outputs of the software runs with the stand alone and combined applications of these evaluation measures are presented in the following sections.

GALib, which is used to perform genetic operations, maximizes the defined objective function by default. Therefore, the inverse of the objective function defined in Equation 4.11 has been taken in the computer implementation.

5.4.2.1 The Experiments Concerning the Stand Alone Application of the Complexity Measure (ω_1=1, ω_2=0)

The global minimum of the structural complexity measure should theoretically yield the feasible design shown in Figure 5.10. This design imports thermal energy from an external source without paying any effort to generate it.

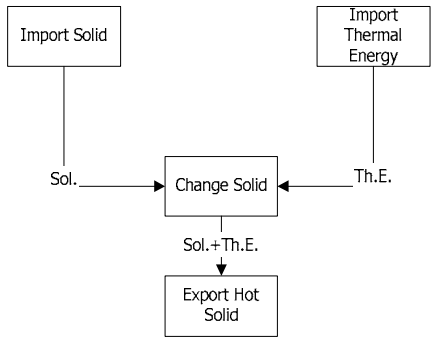


Figure 5.10. The simplest cooker design.

The test-runs performed for the stand alone application of the complexity evaluation measure have converged to this simplest design. However, in some of the runs, it has been observed that, the evolution got stuck in a local minimum representing a feasible but a more complex design. The results of a typical test run, which the evolution converged to the global minimum, are presented in Figures 5.11 and 5.12.

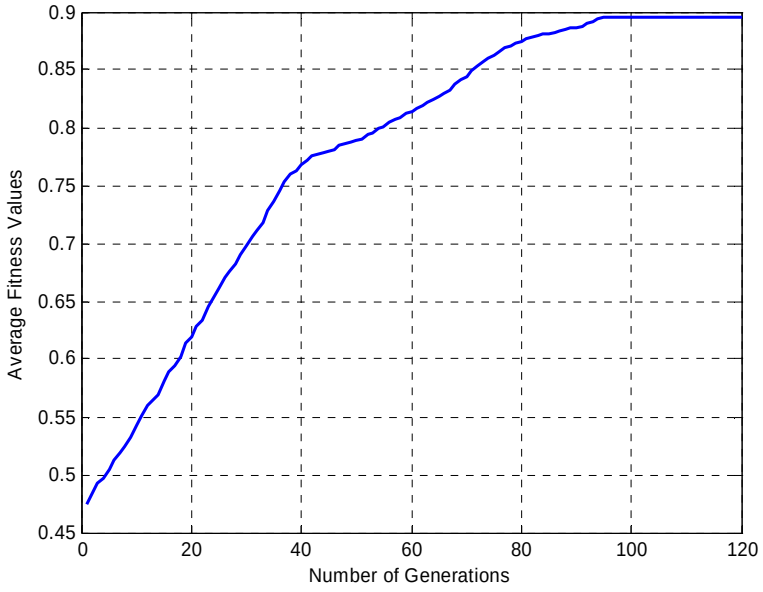


Figure 5.11 The average fitness values of the population throughout the evolution process in the experiment with the weighting coefficients of $\omega_1=1$, $\omega_2=0$.

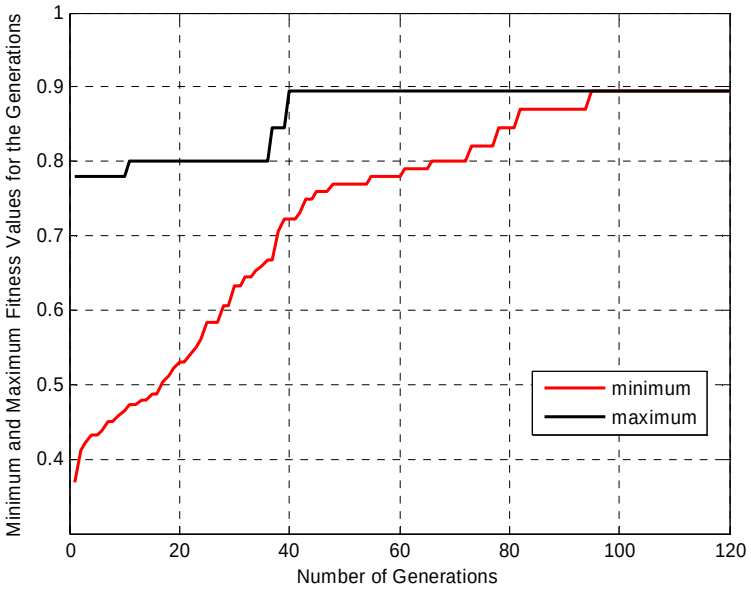


Figure 5.12 The best (i.e. maximum) and the worst (i.e. minimum) fitness values obtained throughout the evolution process in the experiment with the weighting coefficients of $\omega_1=1$, $\omega_2=0$.

The fitness values of the best individual in the initial population monotonically increase as shown in Figure 5.12. This is a result of the steady-state genetic algorithm, which always keeps the best individuals in each generation.

5.4.2.2 The Experiments Concerning the Stand Alone Application of the Creativity Measure (ω_1=0, ω_2=1)

As it is discussed in Section 5.3.2, theoretically the stand alone application of the creativity measure forces the evolutionary process to generate complex individuals due to their low probability of occurrences. The following two experiments have also validated this result.

Experiment 1:

The average fitness values of the population during the evolution process in the experiment 1 are given in Figure 5.13.

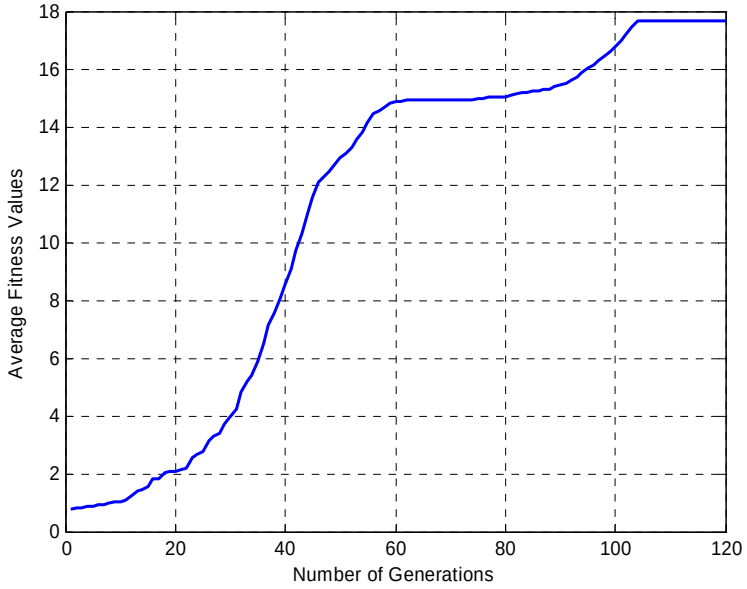


Figure 5.13 The average fitness values of the population throughout the evolution process in the experiment 1 with the weighting coefficients of ω_1=0, ω_2=1.

Figure 5.14 illustrates the maximum and the minimum fitness values of the individuals in each generation.

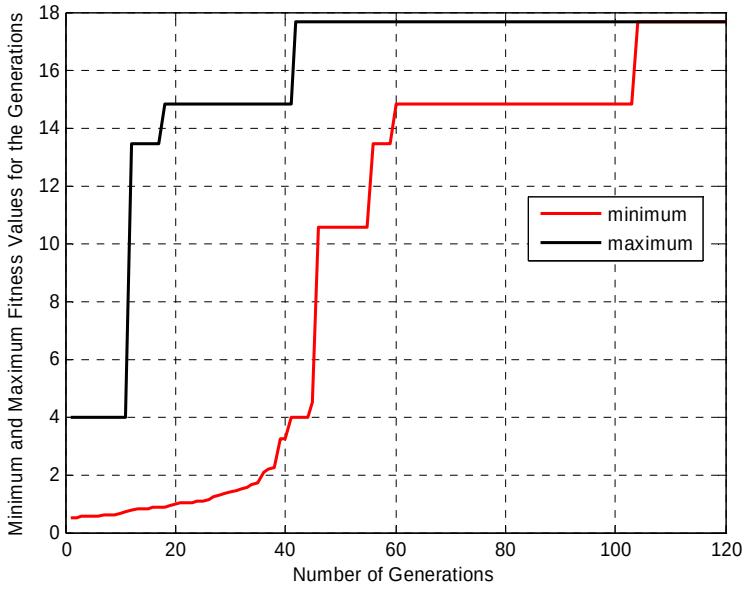


Figure 5.14 The best and the worst fitness values obtained throughout the evolution process in the experiment 1 with the weighting coefficients of ω_1=0, ω_2=1.

In the experiment 1, it was observed that at the end of the generations, the evolution process converged to the individual shown in Figure 5.15. Although the converged solution is a feasible solution, it is very complex and includes unnecessary modules. In this design, the section bordered with a dashed rectangle itself is able to satisfy the design goal. Unlikely, the following part looking like a cooling radiator is completely unnecessary. However, by the addition of this part, the increasing complexity decreases the probability of occurrence of the generated design and so increases the creativity value.

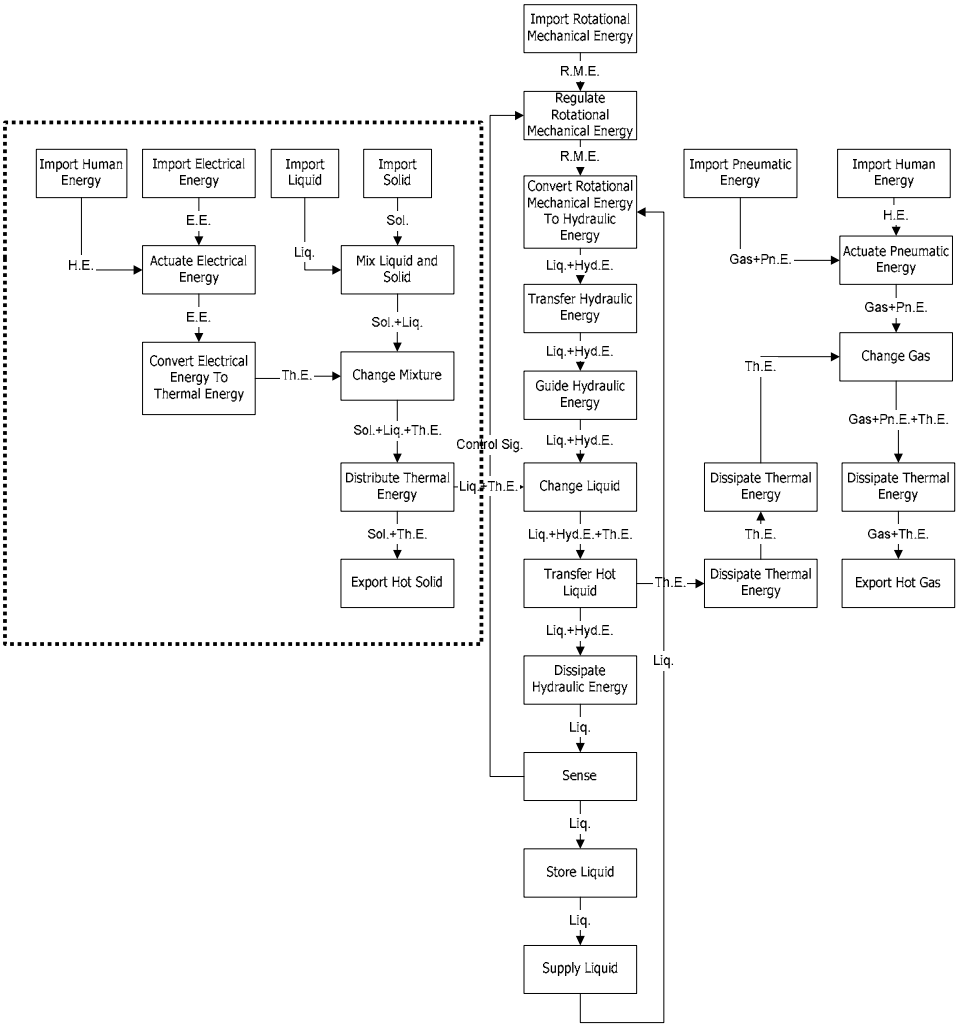


Figure 5.15 The best design alternative generated in the experiment 1 with the weighting coefficients of ω_1=0, ω_2=1.

Experiment 2:

In the experiment 2, the average fitness values of the population and the minimum-maximum fitness values encountered during the generations are presented in Figure 5.16 and 5.17 respectively. The best individual found at the end of the process is given in Figure 5.18.

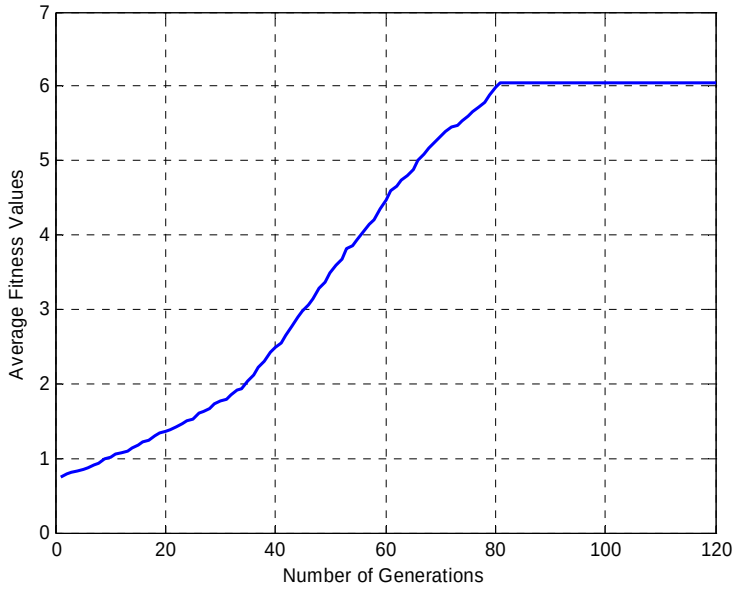


Figure 5.16 The average fitness values of the population throughout the evolution process in the experiment 2 with the weighting coefficients of ω_1=0, ω_2=1.

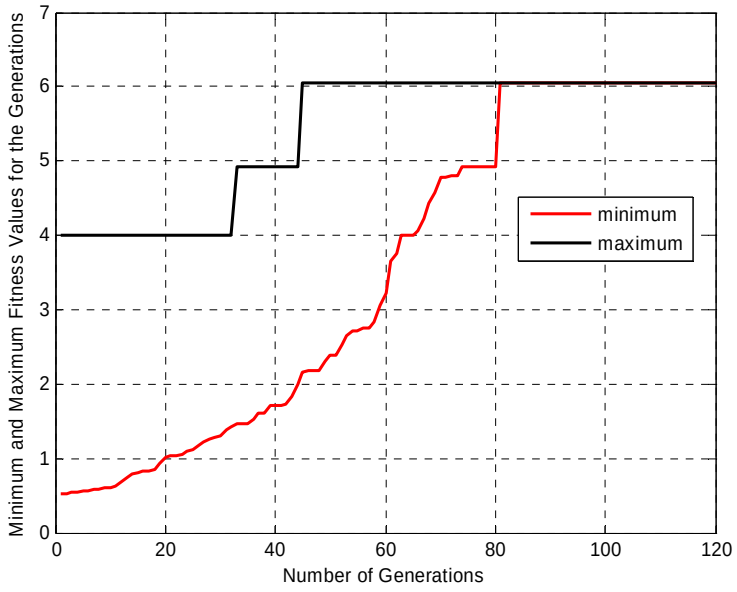


Figure 5.17 The best and the worst fitness values obtained throughout the evolution process in the experiment 2 with the weighting coefficients of ω_1=0, ω_2=1.

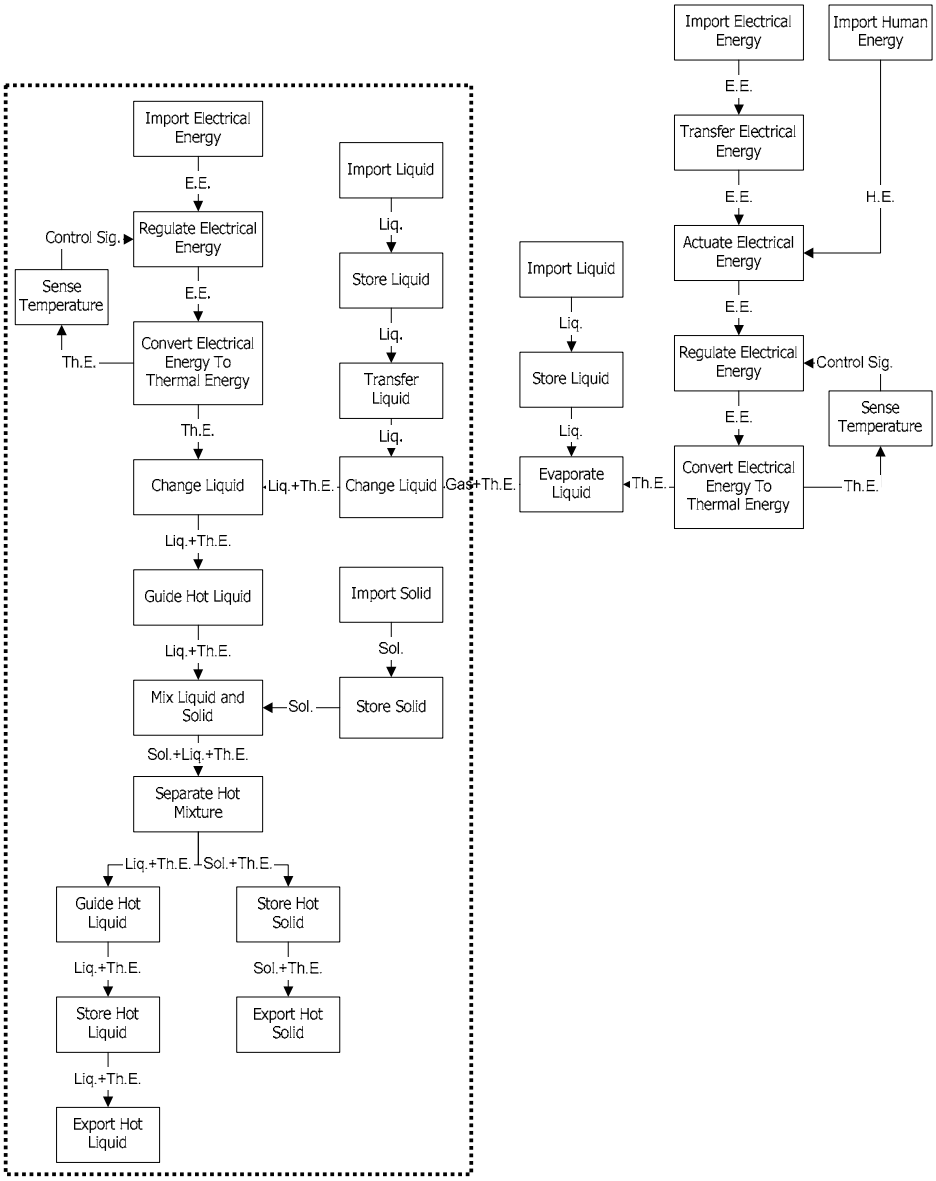


Figure 5.18 The best design alternative generated in the experiment 2 with the weighting coefficients of ω_1=0, ω_2=1.

In Figure 5.18, the section in the dashed rectangle introduces a boiler design and this part is sufficient to fulfill the design goal. The remaining part resembling an evaporator only increases the complexity of the generated individual.

5.4.2.3 The Experiments Concerning the Combined Application of the Complexity and the Creativity Measures (ω_1=0.5, ω_2=0.5)

In order to prevent designs from having unnecessarily complex structures in the generation of creative solutions, the combined application of the complexity and the creativity measures is required. This fact is shown in the following experiments.

Experiment 1:

The best individual generated during the experiment 1 is a boiler working with solar energy as shown in Figure 5.19. It is observed that this individual is composed of an energy conversion module, some parts from the photo-voltaic cell and the rice cooker.

The changes in the average fitness values of the population during the generations are presented in Figures 5.20. Moreover, Figure 5.21 illustrates the fitness values of the best and the worst individuals at each generation.

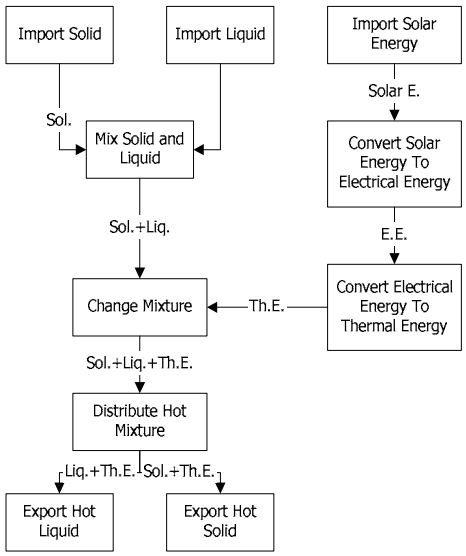


Figure 5.19 The best design alternative generated in the experiment 1 with the weighting coefficients of ω_1=0.5, ω_2=0.5.

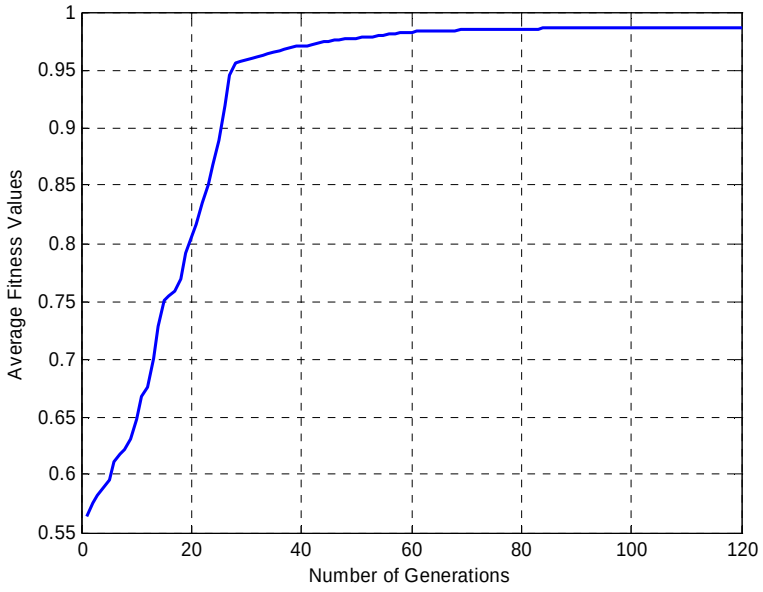


Figure 5.20 The average fitness values of the population throughout the evolution process in the experiment 1 with the weighting coefficients of ω_1=0.5, ω_2=0.5.

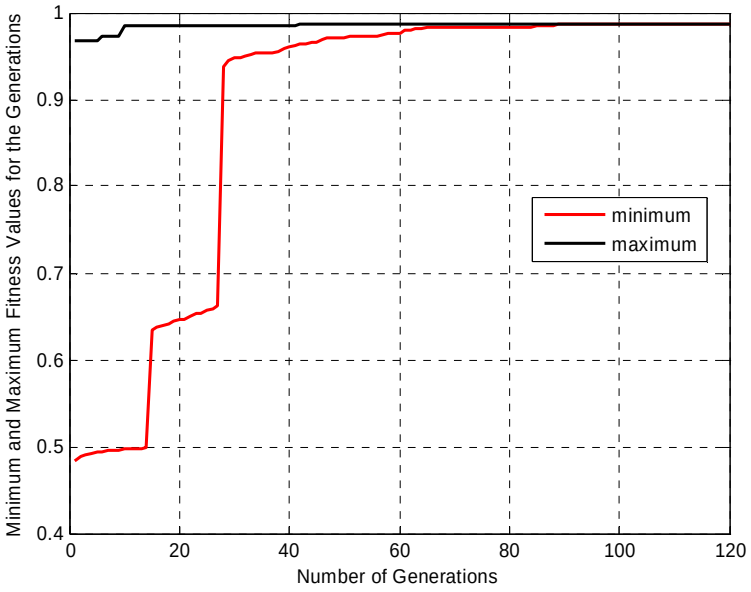


Figure 5.21 The best and the worst fitness values obtained throughout the evolution process in the experiment 1 with the weighting coefficients of ω_1=0.5, ω_2=0.5.

Experiment 2:

In the experiment 2, the best individual is again a strange kind of boiler as shown in Figure 5.22. It generates thermal energy by converting the imported rotational mechanical energy. Although this type of conversion is usually encountered as a type of energy loss in the household appliances, it may be likened to the energy conversion observed in the spin or the friction welding processes.

The average fitness values of the population throughout the evolution process are given in Figure 5.23. Similarly, the best and the worst fitness values obtained throughout the evolution process can be observed in Figure 5.24.

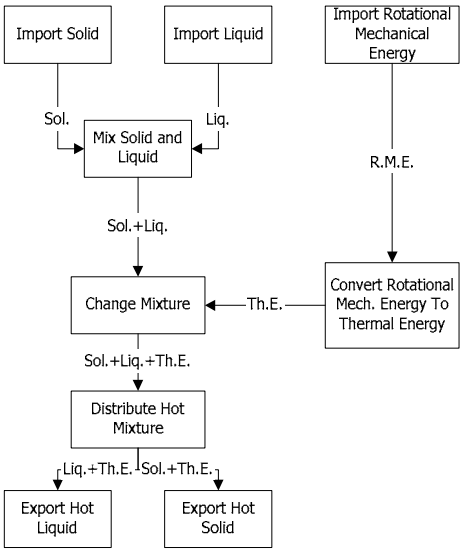


Figure 5.22 The best design alternative generated in the experiment 2 with the weighting coefficients of ω_1=0.5, ω_2=0.5.

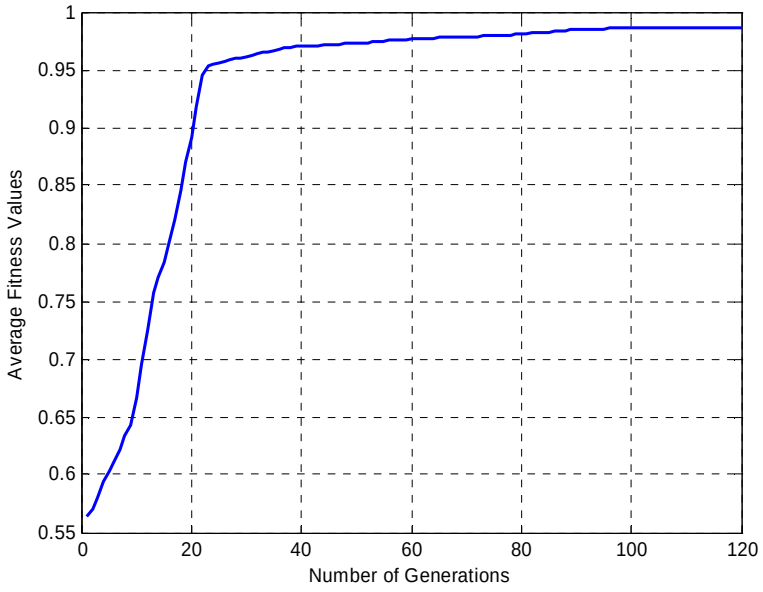


Figure 5.23 The average fitness values of the population throughout the evolution process in the experiment 2 with the weighting coefficients of ω_1=0.5, ω_2=0.5.

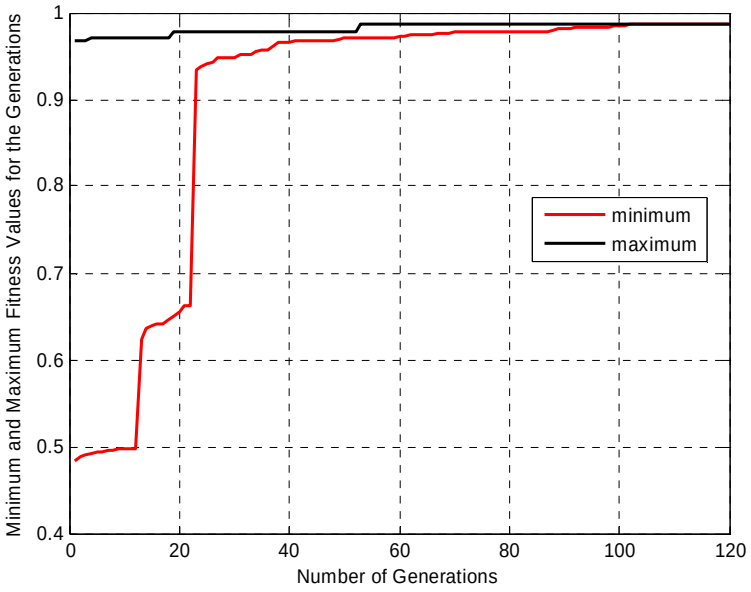


Figure 5.24 The best and the worst fitness values obtained throughout the evolution process in the experiment 2 with the weighting coefficients of ω_1=0.5, ω_2=0.5.

Experiment 3:

In the experiment 3, a cooker similar to the simplest design in Figure 5.10 is generated as the best individual. This design also imports thermal energy without paying any effort to generate it. However, instead of applying this thermal energy to the raw food, the stored liquid is heated and the food is cooked by hot liquid as shown in Figure 5.25. The Figures 5.26 and 5.27 indicates the changes in the average fitness values of the population and the fitness values of the best and worst individuals throughout the process respectively.

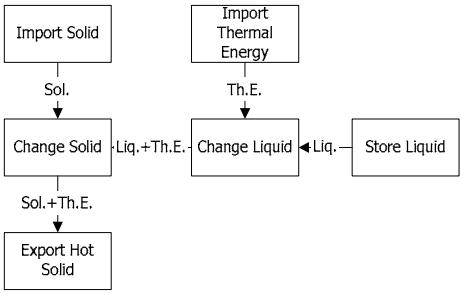


Figure 5.25 The best design alternative generated in the experiment 3 with the weighting coefficients of ω_1=0.5, ω_2=0.5.

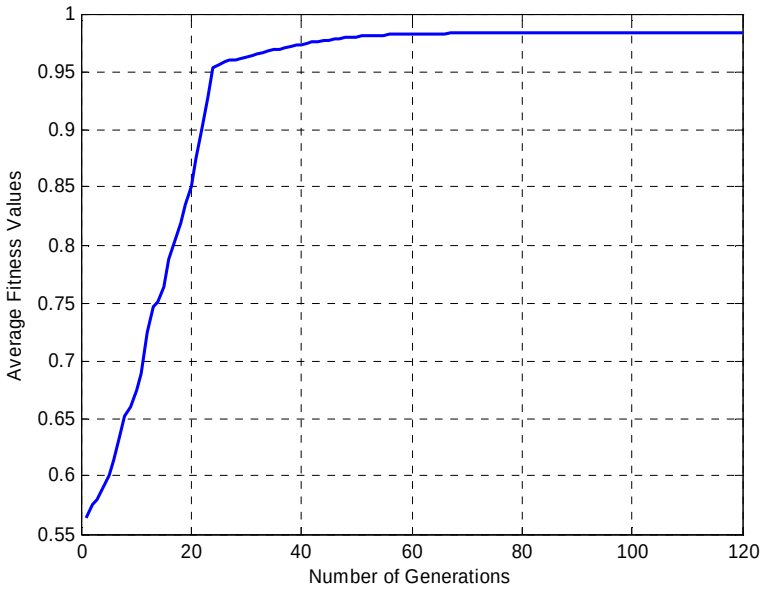


Figure 5.26 The average fitness values of the population throughout the evolution process in the experiment 3 with the weighting coefficients of ω_1=0.5, ω_2=0.5.

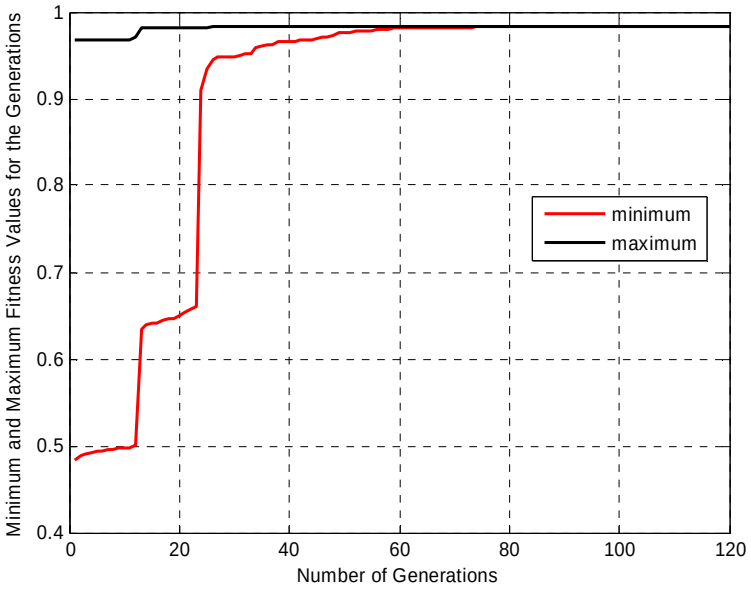


Figure 5.27 The best and the worst fitness values obtained throughout the evolution process in the experiment 3 with the weighting coefficients of ω_1=0.5, ω_2=0.5.

Experiment 4:

In the experiment 4, the best individual generated throughout the evolution process works with chemical energy as shown in Figure 5.28. This design consists of an energy conversion module and the parts from the electric generator, the electric wok.

The tendency of average fitness values of the population during the evolution process is illustrated in Figure 5.29. The fitness values of the best and the worst individuals are also presented in Figure 5.30.

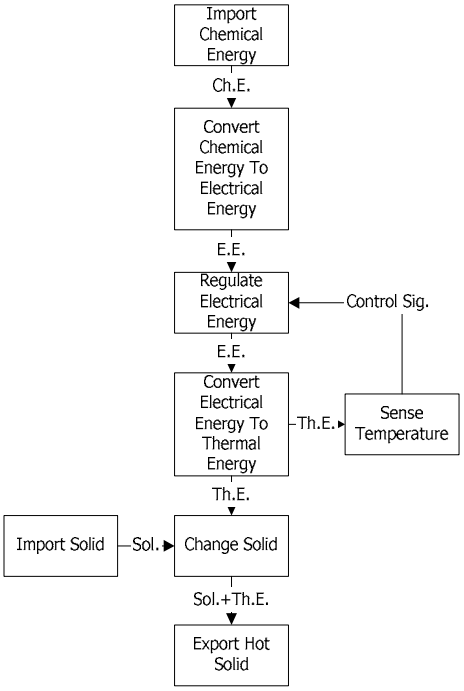


Figure 5.28 The best design alternative generated in the experiment 4 with the weighting coefficients of ω_1=0.5, ω_2=0.5.

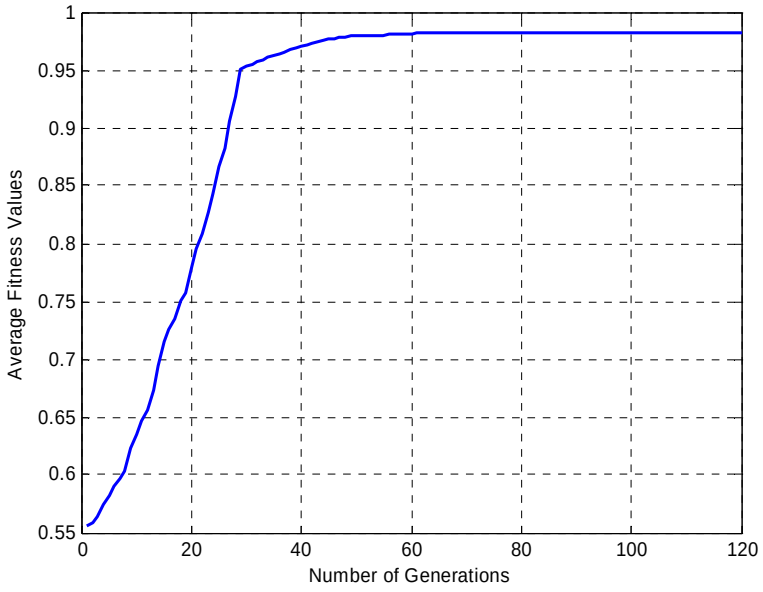


Figure 5.29 The average fitness values of the population throughout the evolution process in the experiment 4 with the weighting coefficients of ω_1=0.5, ω_2=0.5.

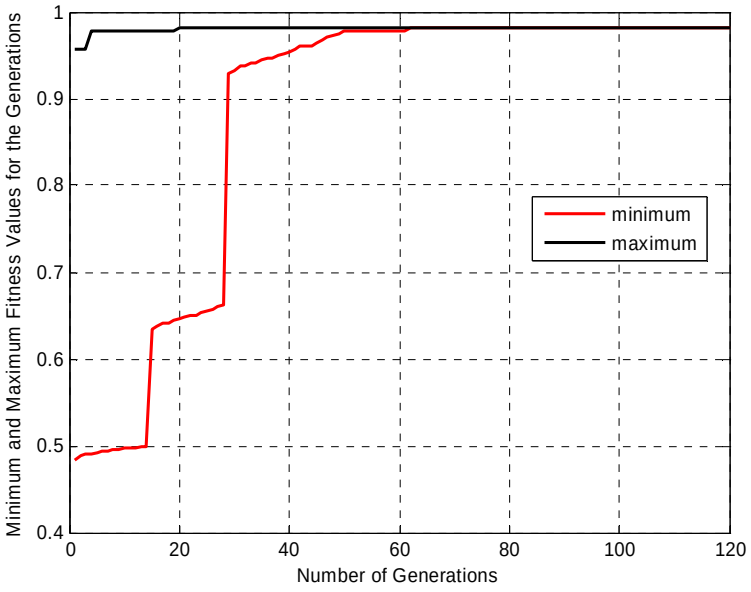


Figure 5.30 The best and the worst fitness values obtained throughout the evolution process in the experiment 4 with the weighting coefficients of ω_1=0.5, ω_2=0.5.

Experiment 5:

In the experiment 5, the best individual resembles a microwave oven working with solar energy as shown in Figure 5.31.

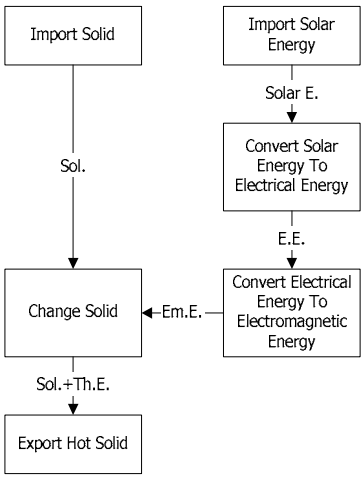


Figure 5.31 The best design alternative generated in the experiment 5 with the weighting coefficients of ω_1=0.5, ω_2=0.5.

The changes of the average fitness values and the fitness values of the best and worst individuals during evolution are presented in Figures 5.32 and 5.33 respectively.

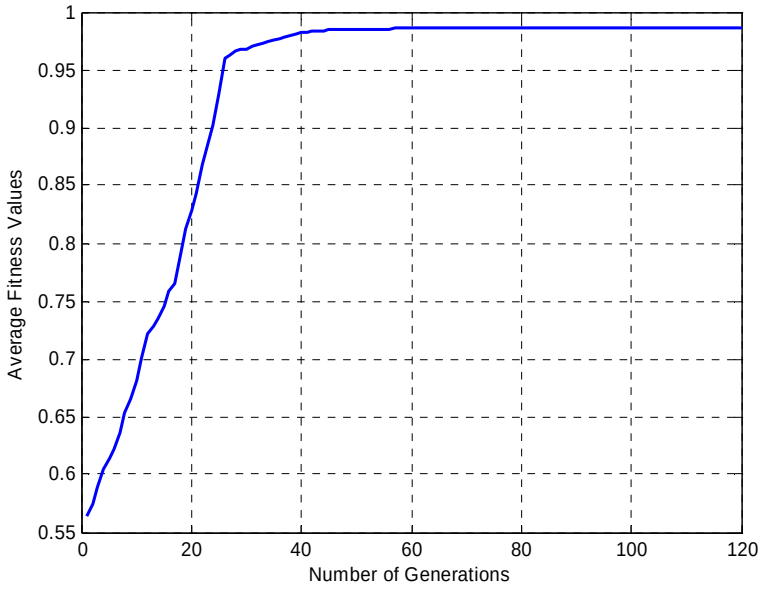


Figure 5.32 The average fitness values of the population throughout the evolution process in the experiment 5 with the weighting coefficients of ω_1=0.5, ω_2=0.5.

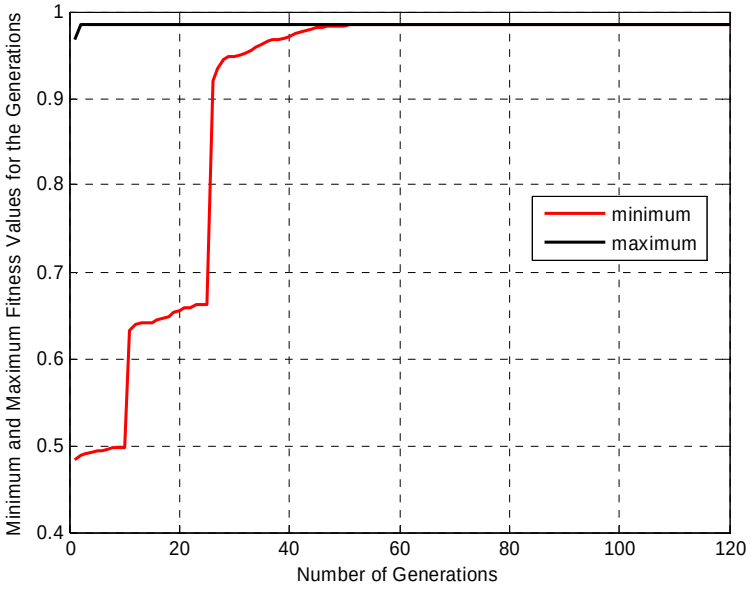


Figure 5.33 The best and the worst fitness values obtained throughout the evolution process in the experiment 5 with the weighting coefficients of ω_1=0.5, ω_2=0.5.

During the test runs, some interesting feasible individuals are also observed in the intermediate generations. Although the evolutionary process does not converge to these individuals, it is beneficial for the designer to investigate them. Some of these individuals arising in the intermediate generations of the experiments are presented below.

In Figure 5.34, a feasible design alternative, which has the similar operating principles with a steam cooker, is presented. In this design, thermal energy is used for producing steam and cooking process is performed by the produced steam.

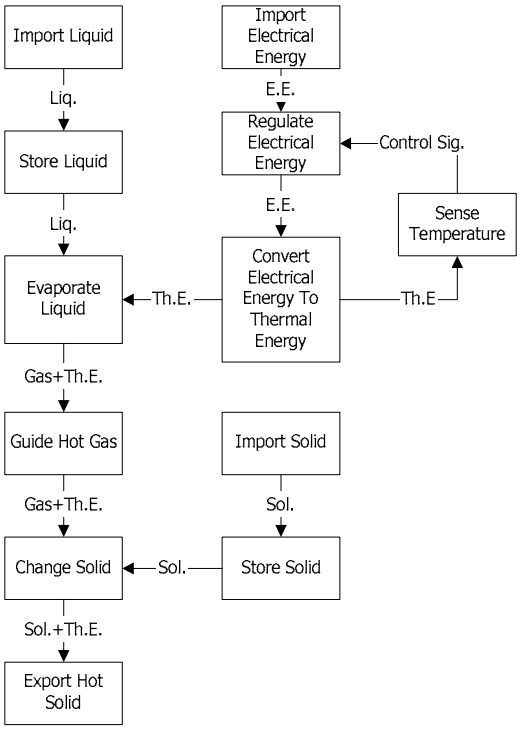


Figure 5.34 A steam cooker arising in the intermediate generations of the experiments with the weighting coefficients of ω_1=0.5, ω_2=0.5.

A popcorn popper working with solar energy is another feasible design alternative arising in the intermediate generations. This design uses hot gas to cook the raw food as shown in Figure 5.35.

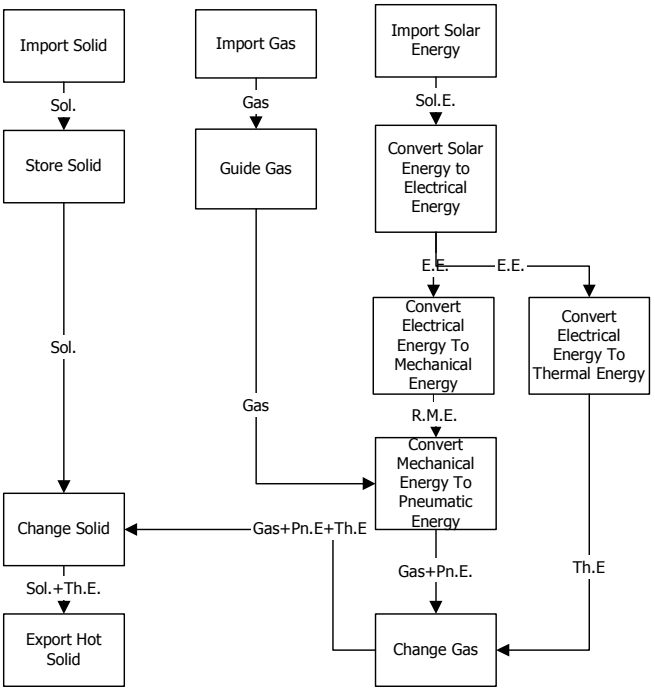


Figure 5.35 A popcorn popper arising in the intermediate generations of the experiments with the weighting coefficients of ω_1=0.5, ω_2=0.5.

A press toaster-like design shown in Figure 5.36 is also a feasible solution alternative generated in the intermediate steps of the evolution.

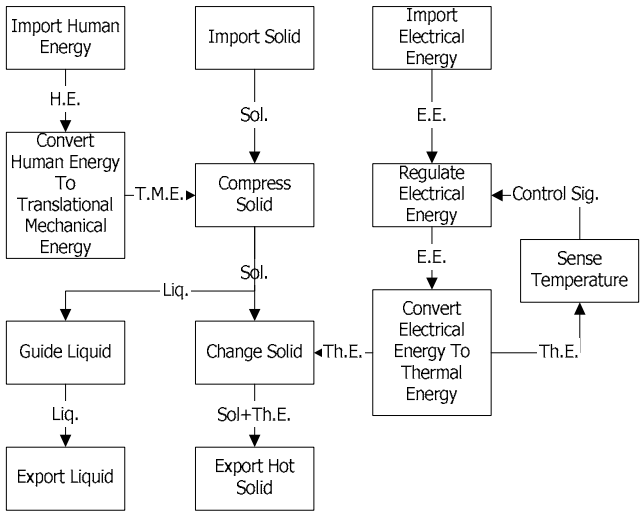


Figure 5.36 A toaster arising in the intermediate generations of the experiments with the weighting coefficients of ω_1=0.5, ω_2=0.5.

In the experiments, two different types of boiler have also been observed. The first type boiler shown in Figure 5.37 heats water and mixes the hot water and the raw food.

The second type boiler illustrated in Figure 5.38 has a more interesting working principle. It produces steam and uses it to control the temperature of the hot water as in the case of the espresso machine. In this design, the hot water is not applied to the raw food, unless the sufficient amount of steam is stored.

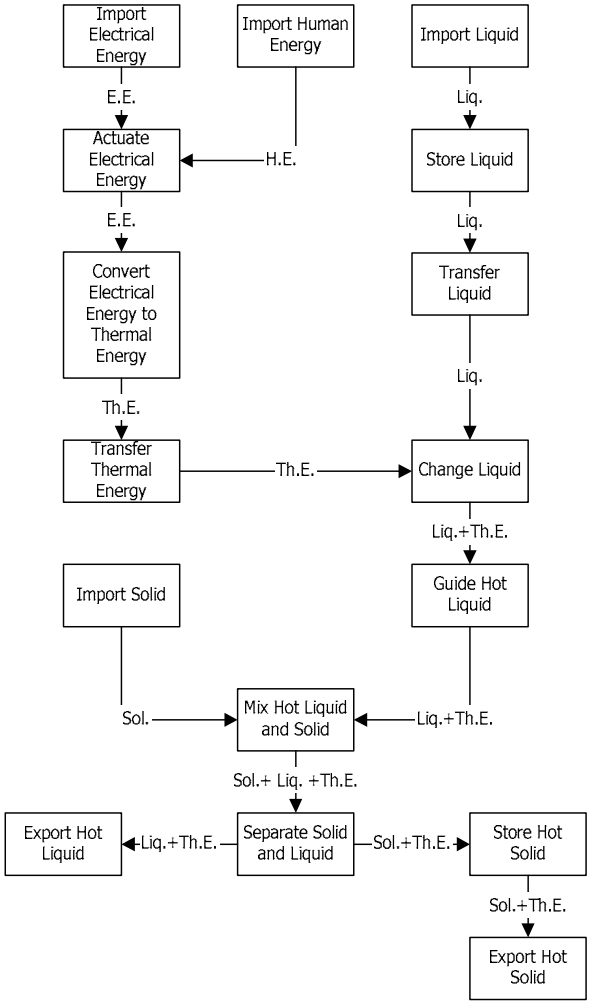


Figure 5.37 The first type boiler arising in the intermediate generations of the experiments with the weighting coefficients of ω_1=0.5, ω_2=0.5.

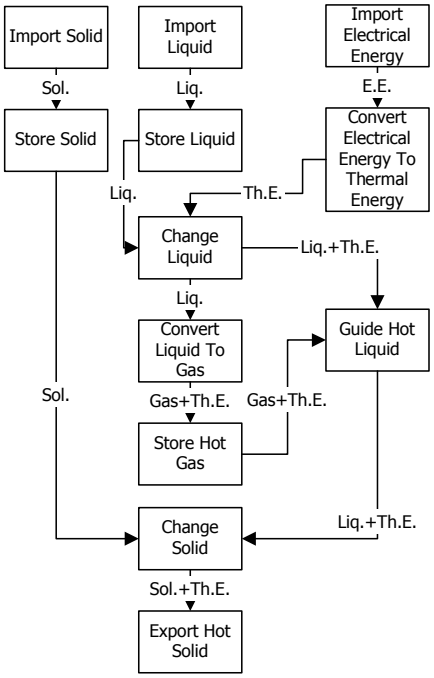


Figure 5.38 The second type boiler arising in the intermediate generations of the experiments with the weighting coefficients of ω_1=0.5, ω_2=0.5.

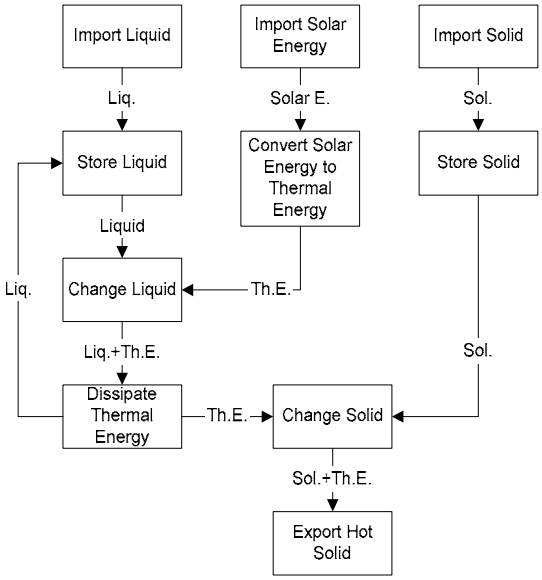


Figure 5.39 A solar cooker arising in the intermediate generations of the experiments with the weighting coefficients of ω_1=0.5, ω_2=0.5.

In Figure 5.39, an interesting solar cooker generated by evolutionary process is presented. Although it utilized hot water to cook the raw food, it does not work like a boiler. The operating principle of this cooker is more similar to a radiator. The hot water and the raw food are never mixed, only a heat exchange occurs between them.

Another strange type of cooker is given in Figure 5.40. In this design, two different solid foods are coupled and pierced (or squeezed) before cooking. The cooking process involved in this design is the same as the microwave oven.

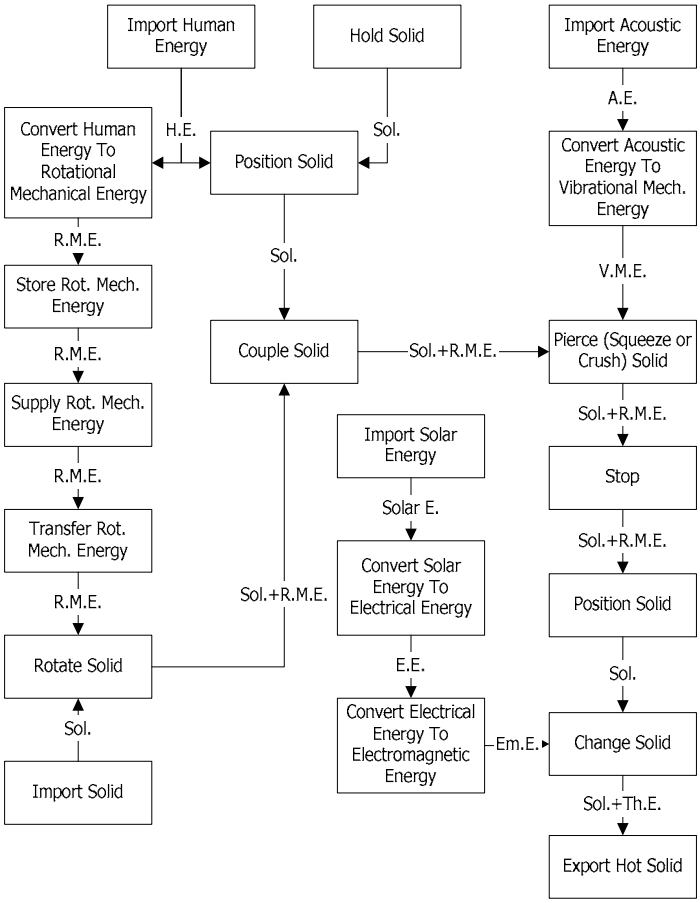


Figure 5.40 A strange type of cooker arising in the intermediate generations of the experiments with the weighting coefficients of ω_1=0.5, ω_2=0.5.

CHAPTER VI

DISCUSSION AND CONCLUSIONS

6.1 SUMMARY AND DISCUSSIONS

The main goal of this thesis is the development of an effective methodology to generate creative solutions for the design tasks without binding solution spaces with the designers' individual experiences and prejudices. Therefore an evolutionary methodology for the functional level conceptual design of engineering products has been proposed.

The 20th century French mathematician Hadamard defines invention or discovery, whether in mathematics or anywhere else, as the novel combination of old ideas (Goldberg, 2004). This statement clarifies this work with its two aspects. First, the combination might result in better than notion taken individually. Second, to produce a creative solution, this combination should be performed on ideas instead of means. From engineering point of view, both of them bring new and important aspects into traditional design approach.

The human way of thinking in combination of ideas stemming from knowledge and experiences to generate creative or innovative solutions is a very intricate subject to understand. This is still the primary goal of many ongoing research studies in the area of cognitive science. Therefore, in order to bring creativity into foreground in the design activities, the proposed methodology employs evolutionary algorithms, which are search procedures based on the mechanics of natural genetics and selection (Goldberg, 1997). Although it is a search

procedure rather than a reasoning technique, it might still set an analogy between evolutionary design process and designer's activities. As the genetic operations: crossover and mutation take on the designer's activities to perform combination and modification of the existing solutions, the evaluation procedure forces the process to act similar to the designer's behavior following his/her inference.

In the literature, there exist many studies adopting evolutionary process in design of engineering artifacts, some of which are mentioned in Chapter 2. All of these studies operate evolutionary algorithms at physical level because of the habits arising form the traditional engineering approach (i.e. bottom-up design). Although these applications become fruitful especially in design optimization tasks, they usually generate functionally equivalent variants. Therefore, in order to generate differences far beyond the physical formation, this study operates evolutionary algorithms at functional level.

In the literature, the problems of accomplishing the conceptual design process completely at functional level are addressed as:

- Its need for a finite, distinct and complete function set to cover all operations, which are performed by the artifacts.
- The lack of a reasoning technique, which does not violate the law of vertical causality, in searching for the form independent solutions.
- Difficulties encountered in evaluating generated solutions at abstract levels of the design process.

In order to overcome the first difficulty, the proposed methodology utilizes a functional basis as a vocabulary. This basis aims to achieve a repeatable and meaningful representation way for the artifacts without relying on the physical structure. The second difficulty was defeated by abandoning hierarchical representation (i.e. representation in tree form) of the design task. Through the atomic function and flow definitions of vocabulary, it becomes possible to represent artifacts at a single level of abstraction (i.e. representation in graph

form) without need for further decomposition. Finally, to be able to compare artifacts at functional level, two evaluation measures, the *complexity* and the *creativity* were proposed.

Neither the creativity nor the complexity is appropriate for being a standalone objective function of the evolutionary design process. The creativity measure drives the evolutionary process to unpredictable combinations of individuals in the initial population through the crossover operation. The design constraints make these combinations creative design alternatives, which is the main goal of the evolutionary design process. This measure is based on the assumption that the feasible designs having unusual operating principles are creative. However, it should be noticed that the complexity is another factor originating unusual designs. Therefore, in order to lead the design activity to genuine feasible solution alternatives without any doubt, another measure managing the complexity of solutions is required. The combined use of the complexity and the creativity measures generate inventive alternatives for the design tasks.

The value indicated by the complexity measure expresses the difficulty of achieving of a design task. Although in many times the design activities starting with complex functional designs conclude with the products having complex forms, it is not a direct measure of the form complexity. In the evolutionary design, the degree of complexity of a generated artifact is not the sole concern of the designer. Instead, this measure is employed primarily for its supportive attribute to the creativity measure.

In addition to the aforementioned difficulties in development of such a methodology, some others have also been encountered throughout the study. First of all, the need for a generic representation scheme of evolutionary process is satisfied by employing the directed graphs. Unlike the string based or the tree based representation schemes, this representation seems working well in revealing interconnections and operation sequence besides the

identification of the required functions. This representation is also eligible for the definition of the genetic operations and the computer implementation.

Additionally, engineering design unavoidably requires dealing with a significant number of problem specific constraints. Although evolutionary algorithms operate successfully in unconstrained environments, some constraint handling techniques are needed to produce feasible alternatives for the constrained optimization problems. In the implementation of the evolutionary design process, it was observed that the static penalty is able to cope with the diversity problem that might be encountered in the initial population. The performed tests show that even if there is only a single feasible individual in the initial population, the best individual generated during the evolution is always a feasible individual. The coefficients of static penalty are optimized by performing many test-runs according to the design problem and the employed initial population.

The presence of several objectives is typical for engineering design problems. Two evaluation measures are presently used in the evolutionary design process. In order to accomplish multi-criteria optimization, the weighted sum method is employed. This method is required to carry out the synthesis with different objective function weights in a systematic way to have a complete understanding of the design task at hand (Cetin, 2003). In the evolutionary design, the weights are determined by performing numerous test runs.

6.2 CONTRIBUTION OF THE THESIS

This is the first study that implements the evolutionary approach to the conceptual design phase. The main contribution of the evolutionary design methodology is to propose a tool to automate the most critical and ambiguous stage of the design process. In the traditional approach, the designer accomplishes this stage with his/her experience and intuition, lacking immediate feedback about his/her design decisions. Automation by the

proposed methodology decreases the dependency of the design activity on the designer. Therefore, it increases the quality of generated designs and the efficiency of the design process. It also presents a systematic design approach for less or inexperienced designers and facilitates a base for experienced designers to conceive the other solution alternatives beyond their experiences.

The developed methodology proposes a set of feasible functional designs to the designer at the end of the conceptual design phase. Each functional design may have numerous physical solution alternatives at the embodiment design phase. Thus, the designer's freedom is kept throughout the design process. It is important especially for the form optimization of the artifact.

Automation in the generation of design alternatives, which is a tedious task for designers, facilitates the designer to concentrate on the evaluation issues only. Hence, the development of new evaluation measures, the constraint handling and the multi-criteria optimization techniques become the primary concerns of designer.

6.3 THE DIFFICULTIES IN THE OPERATION OF THE EVOLUTIONARY DESIGN PROCESS

Besides all these merits, there are also some difficulties in the application of the evolutionary design methodology.

- Besides the best individuals found at the end of the design process, the individuals generated at the intermediate steps of the evolution are also very important. Since, the thousands of feasible individuals are generated during the evolution, the overlooking of the important part of these designs is one of the major difficulties in the operation of evolutionary design process.

- The generated individuals may require interpretations to conceive its operating principles. The designers should avoid getting stuck in the known physical solution alternatives in interpreting the generated designs.

- Since no repair algorithm is operated, the generated individuals may require some modifications.

- Working on non-uniform initial populations limits the effectiveness of the evolutionary process. Moreover, the diversity in the initial population is important for the constrained handling. Therefore, the definition of the initial population is a critical issue influencing the result substantially.

- The evolutionary methodology requires problem specific multi-objective optimization and constraint handling techniques. These techniques determine both the efficiency and the reliability of the process. Therefore, the evolutionary design methodology needs designers knowledgeable about these techniques to operate efficiently.

- In the traditional engineering design activities, designers intuitively avoid some possible impractical design alternatives. All criteria to distinguish these alternatives from viable designs should be defined in the evolutionary design process, otherwise impractical solutions becomes inevitable. In some applications, the impractical solutions are eliminated by employing interactive evolutionary algorithms.

- It should be noticed that, many of the today's consumer products have been developed by designers throughout their historical development period. This period acts as a natural evolution process, which leaves a little to the evolutionary design process. Therefore, finding creative solutions for these products every time is impossible to achieve for the evolutionary design process.

- Finally, in computer implementation, the size of generated artifacts is limited due to the increasing computational cost for larger artifacts. Although this limitation is not a problem for the test case of the household appliance design, this may be a problem in generation of more complicated artifacts.

6.4 SUGGESTIONS FOR FUTURE WORK

The evolutionary design methodology has many promising paths for future research; some of them are given below.

- Expanding the Initial Population: The performed test runs indicated that especially feasible individuals in the initial population have a significant effect on the generated designs. Therefore, the initial population should be expanded through increasing the feasible individuals.

- Application of the evolutionary design process to different design tasks: In this thesis, the evolutionary design process has been applied to the design of a household cooker, which is a very difficult task to find innovative solutions due to its long historical improvement background. Although, this case study has also resulted in different solution alternatives, working on the design topics, which are more open to the advances in technology, such as automotive engineering or aerospace engineering, may generate more innovative design solutions.

- Definition of a Multipoint Crossover Operation: Empirical studies performed on genetic algorithms employing string based representation showed that multipoint crossover operation increases the effectiveness of evolutionary algorithms on some problems. Although the graph based representation requires a more complicated

procedure to perform multipoint crossover, such an operation might result in the generation of surprising design alternatives.

- Definition of Other Evaluation Measures: In order to compare the generated design alternatives, depending on the design problem numerous evaluation measures can be proposed. Some typical evaluation measures employed in traditional design activities are cost, quality, reliability, maintainability, performance, ease of use, redundancy, aesthetics, safety, compatibility with other systems and effect on environment etc. However, many of these criteria can not be adopted due to their dependency on the form of the artifact (e.g. reliability, cost etc.). The measures of evolutionary design process might be increased through redefining these general measures at functional level or developing new problem specific criteria.

- Implementation of Other Multi-objective Optimization Methods: Weighted sum approach is the simplest and probably the most common way of handling multi-objective optimization tasks. In order to increase the effectiveness of the design process, other methods can also be applied. However, GALib does not support other multi-objective optimization methods. Therefore, computer implementation for these methods necessitates the utilization of other available software packages such as Open BEAGLE or using custom libraries.

- Implementation of Other Constraint Handling Techniques: In the literature, there exist many variations of the penalty functions and other constraint handling techniques such as immune system emulation or ant colonies. Although some of these techniques such as repair algorithms seem inappropriate for the evolutionary design process, the computer implementation of some promising techniques might be done as a future work.

- Computer Implementation for Parallel Processing: Evolutionary algorithms are eligible for parallel processing. Populations and/or individuals can be evolved in parallel. Parallel processing significantly decreases the evolution time, therefore it allows increasing the number of individuals in the initial population or maximum number of generations. Especially for the designs of complicated artifacts, parallel processing is essential.

- Promoting Modularity In Evolutionary Design: Some highly repetitive modules included in individuals can be encapsulated in evolutionary process. These modules can not be deformed by genetic operations. This decreases the size of individuals and hence the computational costs.

- Determination of Strategies in Education of Designers: This novel design methodology necessitates designer to be well-informed on some specific topics such as criteria in generation of initial populations, handling constraints and multiple objectives in evolutionary algorithms etc. Therefore, some strategies are required to educate designers on the evolutionary design methodology.

- Extension of Evolutionary Methodology to Embodiment Design: In the literature, there exist many evolutionary design applications performed at physical level. By adopting one of them, this study might be extended to cover all stages of design process up to prototyping phase.

REFERENCES

Akiyama, K., 1991, *Function Analysis: Systematic Improvement of Quality Performance*, Productivity Press.

Al-Hakim, L., Kusiak, A. and Mathew, J., 2000, "A Graph-Theoretic Approach to Conceptual Design with Functional Perspectives", *Computer Aided Design*, Vol. 32, pp. 867-875.

Allen, R., Sriram, R., Szykman, S. and Fijol, R. J., 2000, "Representing an Artifact Transport System in a Design Repository: A Case Study", *Proceedings of the ASME Design Engineering Technical Conference & Computers & Information in Engineering Conference*, Maryland, USA.

Altshuller, G., 1984, *Creativity as an Exact Science*, Gordon and Branch Publishers, New York, USA.

Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T. A., Ho, P., Nicollin, X., Olivero, A., Sifakis, J. and Yovine, S., 1995, "The Algorithmic Analysis of Hybrid Systems", *Theoretical Computer Science*, Vol. 138, pp.3-34.

Amos, D., Savinder D., Ujval, A. and Sameer, A., 2001, *Axiomatic Design Theory*, Lecture Notes, University of Maryland, Maryland, USA.

Amerogen, J. van, 2000, "Modelling, Simulation and Controller Design for Mechatronic Systems with 20-sim 3.0", *Proceedings 1st IFAC Conference on Mechatronic Systems*, Darmstadt, Germany, pp. 832-836.

Anderson, J., 2001, *A Survey of Multi-Objective Optimization in Engineering design*, Technical Report, Linköping University, Linköping, Sweden.

Andreasen, M. M., 1980, *Machine Design and Development*, Ph.D. Dissertation, Lund University, Lund.

Bahrami, A. and Dagli, C. H., 1994, "Design Science", *Intelligent Systems in Design and Manufacturing*, Editors: C. H. Dagli and A. Kusiak, ASME Press, New York, pp. 7-25.

Bently, P. J. and Wakefield J. P., 1996, "Conceptual Evolutionary Design by Genetic Algorithm", *Engineering Design and Automation 2:3*, John Wiley and Sons Inc.

Blanchard, B. S. and Fabrycky, W. J., 1998, *Systems Engineering and Analysis*, 3rd Ed., Prentice Hall, New Jersey, USA.

Blamire, J., 1997, *Genotype and Phenotype Definition*, Science at a Distance, http://www.brooklyn.cuny.edu/bc/ahp/BioInfo/GP/Definition.html.

Blickle, T. and Thiele, L., 1995, *A Comparison of Selection Schemes used in Genetic Algorithms*, TIK Report, Computer Engineering and Communication Networks Lab., Swiss Federal Institute of Technology, Zurich, Switzerland.

Boden, M. A., 1991, *The Creative Mind: Myths and Mechanisms*, Basic Books, New York, USA.

Bracewell, R. H. and Sharpe, J. E. E., 2004, "Functional Descriptions used in Computer Support for Qualitative Scheme Generation: Schemebuilder", *Schemebuilder: Computer Aided Knowledge Based Design of Mechatronic Systems*, http://www.comp.lancs.ac.uk/edc/publications/automation.pdf.

Breunese, A. P. J. and Broenink, J. F., 1997, "Modeling Mechatronic Systems Using the SIDOPS+ Language", *ICBGM '97*, Phoenix, AZ, USA, pp. 301-306.

Broenink, J, F., 1999, "Introduction to Physical Systems Modeling with Bond Graphs", *SiE Whitebook on Simulation Methodologies*, http://www.rt.el.utwente.nl/bnk/papers/BondGraphsV2.pdf.

Broenink, J. F. and Kleijn, C., 1999, "Computer-Aided Design of Mechatronic Systems Using 20-Sim 3.0", *Proceedings WESIC'99, 2nd Workshop on European Scientific and Industrial Collaboration*, Newport, UK, pp. 27-34.

Broenink, J. F., 1997, "Bond Graph Modeling in Modelica", *European Simulation Symposium*, Passau, Germany, Oct. 19-22.

Burton, A., 2004, "Truss Optimization using Genetic Algorithms", *Proceedings of Genetic and Evolutionary Computation Conference*, GECCO, Seattle, Washington, USA.

Buur, J., 1990, *A Theoretical Approach to Mechatronics Design*, Ph.D. Dissertation, Technical University of Denmark, Lyngby, Denmark.

Carlson, S. E. and Shonkwiler, R., 1998, "Annealing a Genetic Algorithm over a Constraint", *Proceedings of SMC98: IEEE International Conference on Systems*, San Diego, USA.

Carruba, F. P., 1993, "Designing People Pleasing Products", *9th international Conference on Engineering Design, ICED'93*, Heurista, Zurich, Switzerland.

Cetin, O., 2003, *Decomposition–Based Assembly Synthesis of Family of Structures*, Ph.D. Dissertation, The University of Michigan, Michigan, USA.

Chandrasekaran, B. and Stone, R., 2001, "An Inductive Approach to Product Design Based on Modular Product Architecture", ASME DETC'01, http://function.basiceng.umr.edu/delabsite/DESite/PDF/DFM-DETC_2001_stone_balaji.pdf.

Chakrabarti, A. and Bligh, T. P., 2001, "A Scheme for Functional Reasoning in Mechanical Conceptual Design", *Design Studies*, 22, 6, pp.493-517.

Chakrabarti, A., 1997, "Supporting Deep Understanding and Problem Solving Using Function Structures: A Case Study", *Proceedings of International Conference on Engineering Design*, ICED97, Tampere, Finland, Vol. 3, pp. 70-76.

Chakrabarti, A., 1993, "Towards a Theory of Functional Reasoning in Design", *Proceedings of the International Conference on Engineering Design*, ICED93, Hague, Netherlands.

Chakrabarti, A. and Bligh, T. P., 1992, "A Knowledge-Based System for Synthesis of Single Input Single Output Systems in Mechanical Concept Design", *Proceedings of EXPERSYS 1992*, Houston, Texas, USA.

Coello, C. A. C., 2002, "Theoretical and Numerical Constraint-Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art", *Computer Methods in Applied Mechanics and Engineering*, Vol. 191, No. 11-12, pp. 1245-1287.

Coello, C. A. C., 2001, "A Short Tutorial on Evolutionary Multi-objective Optimization", *The Proceedings of First International Conference on Evolutionary Multi-Criterion Optimization (EMO'01)*, Zurich, Switzerland, pp. 21-40.

Coello, C. A. C., 2000, "Treating Constraints as Objectives for Single-Objective Evolutionary Optimization", *Engineering Optimization*, 32-3, pp. 275-308.

Coello, C. A. C., 1996, *An Empirical Study of Evolutionary Techniques for Multi-Objective Optimization in Engineering Design*, Ph.D. Dissertation, Tulane University, New Orleans, USA.

Collins, J., Hagan, B., and Bratt, H., 1976, "The Failure-Experience Matrix - a Useful Design Tool," *Transactions of the ASME, Series B, Journal of Engineering in Industry*, Vol. 98, pp. 1074-1079.

Coşkun, Ç, 2004, *Software Development for Multi Level Petri Net Based Design Inference Network*, M.Sc. Dissertation, Mechanical Engineering Department, Middle East Technical University, Ankara, Turkey.

Davis, L., 1991, *Handbook of Genetic Algorithms*, Van Nostrand, Reinhold, New York, USA.

Dixie, L. E., 2004, "Whole Qualification Standard for the National Diploma in Engineering",
http://www.ctech.ac.za/facul/eng/ee/convener/docs/ndipdoc.html.

Downing, C. J., Cork, R. T. C., Byrne, B., Coveney, K. and Marnane, W. P., 1996, "Controller Optimisation and System Identification using Genetic Algorithms", *Proceedings of Irish DSP and Control Colloquium*, pp. 45-52.

Dyer, M. G., Flowers, M. and Hodges J, J., 1986, "EDISON: An Engineerign Design Invention System Operating Naively", *Artificial Intelligence*, Vol. 1, No. 1, pp. 36-44.

Dyer, M. and Flowers, M., 1984, "Automating Design Invention", *Proceedings of the Autofact 6 Conference*, Anaheim, California, USA, pp. 25-1 to 25-21.

Edward, C., Xin, L., Schmidt, L. C., 2000, *The Need for a Form, Function, and Behavior-based Representation System*, Lab. Report,
http://www.enme.umd.edu/DATLab/.

Elmqvist, H. and Mattsson, S. E., 1997, "An Introduction to Physical Modeling Language Modelica", *Proceedings of the 9th European Simulation Symposium, ESS'97*, Oct 19-23, Passau, Germany.

Elmqvist, H., Brück, D. and Otter M., 1996, *Dymola: User's Manual*, Dynasim AB, Research Park Ideon, Lund, Sweden.

Erden, A., 2004, *Engineering Design: Concepts, Tools and Education*, Unpublished Manuscript.

Erden, Z., 1999, *A Petri Net Based Design Inference Network for Design Automation at Functional Level Applied to Mechatronic Systems*, Ph.D. Dissertation, Middle East Technical University, Ankara, Turkey.

Erden Z., Erkmen A. and Erden A., 1999, "Handling Uncertainty In Design Automation Using Intuitionistic Fuzzy Propositions", *International Conference on Engineering Design*, ICED 99, Munich, Germany.

Erden, Z., Erkmen, A. M. and Erden, A., 1996, "Generation of Functional Cells for a Mechatronic Design Network", *Proceedings of the 5th UK Mechatronics Forum International Conference (Mechatronics 96) with the 3rd International Conference on Mechatronics and Machine Vision in Practice (M2VIP 96)*, Guimaraes, Portugal, Vol.1, pp. 233-238.

Feng, S. and Song E. Y., 2000, "Information Modeling on Conceptual Process Planning Integrated with Conceptual Design", *Proceedings of the 5th Design for Manufacturing Conference in The 2000 ASME Design Engineering Technical Conferences*, Paper Number: DETC00/DFM-14009, Baltimore, Maryland, USA, September 10-13.

Fonseca, C. M. and Fleming, P. J., 1993, "Genetic Algorithms for Multi-Objective Optimization: Formulation, Discussion and Generalization", *Genetic Algorithms: Proceedings of the Fifth International Conference*, San Mateo, CA, USA.

Freeman, P. and Newell, A., 1971, "A Model for Functional Reasoning in Design", *Proceedings of the Second International Joint Conference on Artificial Intelligence*, London, UK, pp. 621-633.

Fritzson, P., Engelson,V. and Viklund L., 1993, "Variant Handling, Inheritance and Composition in the ObjectMath Computer Algebra Environment*", Design and Implementation of Symbolic Computation Systems (DISCO)*, Edited By Miola, A.,Springer-Verlag, Volume 722 of LNCS, Gmunden, Austria, pp. 145-160.

Goldberg, D. E., 1997 "The design of innovating machines", *Unpublished Manuscript*.

Goldberg, D. E. and Liepins, G., 1991, "A Tutorial on Genetic Algorithm Theory", *The Fourth International Conference on Genetic Algorithms*, University of California at San Diego, La Jolla, California, USA.

Goldberg, D. E., 1989a, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Co, USA.

Goldberg, D. E., 1989b, "Sizing Populations for Serial and Parallel Genetic Algorithms", *Proceedings of the Third International Conference on Genetic Algorithms*, Edited by Schaffer, J. D., San Mateo, California, USA, pp.70–79.

Goldberg, D. E., Kord, B. and Deb, K., 1989c, "Messy Genetic Algorithms: Motivation, Analysis, and First Results", *Complex Systems*, 3(5): pp. 493-530.

Grabowski, H. and Benz, T., 1991, "Implementing the Design Methodology", *Intelligent CAD III*, Edited by Yoshikawa, H., Arbab F. and Tomiyama T., Elsevier Science Publishers B. V., North-Holland, pp. 109-127.

Güroğlu, S., Erden, A., Akbulut, D. and Özgüç, B., 2005, "Creativity: As an Evaluation Measure for Evolutionary Conceptual Design", *Submitted to the Sixth International Conference on Computer Aided Industrial Design and Conceptual Design*, Delft, Netherlands.

Güroğlu, S. and Erden, A., 2004, "Development of an Evaluation Measure for Genetic Approach to Functional Level Conceptual Mechatronic Design", *The Proceedings of 9th Mechatronics Forum - International Conference*, MECHATRONICS 2004, Ankara, Turkey.

Güroğlu, S. and Erden, A., 2003, "Development of a Representation Scheme for the Application of Genetic Methodology to Functional Design", *Proceedings of International Conference on Engineering Design, ICED'03*, Stockholm, Sweden.

Güroğlu S., 1999, *Implementation of an Algorithm for a Petri Net Based Design Inference Network*, M.Sc. Thesis, Mechanical Engineering Department, Middle East Technical University, Ankara, Turkey.

Harik, G., Cantu-Paz E., Goldberg D. and Miller, B., 1999, "The Gambler's Ruin Problem, Genetic Algorithms, and the Sizing of Populations", *Evolutionary Computation*, Vol. 7-3, pp.231-253.

Hirtz, J., Stone, R., McAdams, D., Szykman, S. and Wood, K., 2001a, "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts", *Research in Engineering Design*, 13(2):65-82.

Hirtz, J., Stone, R., McAdams, D., Szykman, S. and Wood, K., 2001b, "Evolving a Functional Basis for Engineering Design", *Proceedings of DETC2001*, DETC2001/DTM-21688, Pittsburgh, Pennsylvania, USA.

Hobbs, M. H. V. and Rodgers, P. J., 1998, "Representing Space: A Hybrid Genetic Algorithm for Aesthetic Graph Layout", *Proceedings of JCIS'98, The 4th Joint Conference on Information Sciences*, Vol. 2, pp. 415-418.

Holland, J. H., 1975, *Adaptation in Natural and Artificial Systems*. The MIT Press, Cambridge, MA, USA.

Houstis C., 2004, *Conceptual Modeling: An Introduction*, Lecture Notes, http://support.inf.uth.gr/courses/CE428/.

Hubka V., 1976, *Theorie der Konstruktionsprozesse*, Springer-Verlag, Berlin.

Hundal, M., 1990, "A Systematic Method for Developing Function Structures, Solutions and Concept Variants", *Mechanism and Machine Theory*, 25(3), pp. 243-256.

IEEE, 1999, *Standard VHDL Analog and Mixed-Signal Extensions*, Technical Report *IEEE Std. 1076.1*, http://www.designers-guide.com/Modeling/1076.1-1999.pdf.

Jeandel, A., Boudaud, F., Ravier, P. and Bushing A., 1996, "U.L.M: Un Langage de Modélisation, A Modelling Language", *Proceedings of the CESA'96 IMACS Multiconference*, IMACS, Lille, France.

Jochum, P. and Kloas M., 1994, "The Dynamic Simulation Environment-Smile", *The Second Biennial European Conference on System Design & Analysis*, ASME, pp. 53–56.

Joines, J. A. and Houck, C. R., 1994, "On the Use of Non-Stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems with GA's", *Proceedings of International Conference on Evolutionary Computation*, IEEE, Orlando, Florida, USA, pp. 579-584.

Kazarlis, S. and Petridis V., 1998, "Varying Fitness Functions in Genetic Algorithms: Studying the Rate of Increase of the Dynamic Penalty Terms", *Proceedings of the 5th Parallel Problem Solving from Nature (PPSN V)*, Heidelberg, Germany, pp. 211-220, http://www.cs.cinvestav.mx/~constraint/papers/kazarlis98.ps.gz.

Kirschman, C., Fadel, G., 1998, "Classifying Functions for Mechanical Design", *Journal of Mechanical Design, Transactions of the ASME*, 120(3), pp. 475-482.

Klir, J. G. and Folger T. A., *Fuzzy Sets, Uncertainty and Information*, Prentice Hall, New Jersey, USA.

Koch, P., Peplinski, J., Allen, J. and Mistree, F., 1994, "A Method for Design Using Available Assets: Identifying a Feasible System Configuration", *Behavioral Science*, 30, pp. 229-250.

Korkmazel, M., 2001, *Development of Multi Level Petri Net Based Design Inference Network*, M.Sc. Thesis, Middle East Technical University, Ankara, Turkey.

Koza, J. R., 1992, *Genetic Programming*, MIT Press, Cambridge, MA, USA.

Koza, J. R., 1994, *Genetic Programming II*, MIT Press, Cambridge, MA, USA.

Krisp, H. and Müller-Schloer C., 2000, "Towards a High Level System Design Using UML and Java", 3rd International UML Conference-Workshop, York, UK.

Kurfman, M., Rajan, J., Stone, R. and Wood, K., 2001, "Functional Modeling Experimental Studies", *Proceedings of DETC2001, DETC2001/DTM-21709*, Pittsburgh, Pennsylvania, USA.

Kusiak, A., 2000, *Computational Intelligence in Design and Manufacturing*, John Willey and Sons Inc., New York, USA.

Kusiak, A., Szczerbicki, E., Park, K., 1991a, "A Novel Approach to Decomposition of Design Specifications and Search for Solutions", *International Journal on Production Research*, Volume 29, No. 7, pp. 1391-1406.

Kusiak, A., Szczerbicki, E., Vujosevic, R., 1991b, "Intelligent Design Synthesis: An Object Oriented Approach", *International Journal on Production Research*, Volume 29, No. 7, pp. 1291- 1308.

Langdon W. B. and Qureshi A., 1995, *Genetic Programming : Computers using 'Natural Selection' to Generate Programs*, Technical Report, RN/95/76, London, UK.

Lawton, J. and Wipke, T., 1999, "Automatic Molecular Design Using Evolutionary Techniques", *Nanotechnology*, Vol 10, pp. 290-299.

Little, A., Wood, K., and McAdams, D., 1997, "Functional Analysis: A Fundamental Empirical Study for Reverse Engineering, Benchmarking and Redesign", *Proceedings of the 1997 Design Engineering Technical Conferences*, 97-DETC/DTM-3879, Sacramento, California, USA.

Lipson, H., Pollack, J. B. and Suh, N. P., 2001, "Promoting Modularity in Evolutionary Design", *Proceedings of Design Engineering Technical Conference*, DETC'01, Pittsburgh, USA.

Malmqvist, J., Axelsson, R., and Johansson, M., 1996, "A Comparative Analysis of the Theory of Inventive Problem Solving and the Systematic Approach of Pahl and Beitz", *Proceedings of the 1996 ASME Design Engineering Technical Conferences*, 96-DETC/DTM-1529, Irvine, California, USA.

Malmqvist, J., 1995, "A Computer-Based Approach Towards Including Design History Information in Product Models and Function-Means Trees", *Proceedings of DTM-95*, Boston, Massachusetts, USA, pp 593–602.

Mathworks, 2004a, *Genetic Algorithm and Direct Search Toolbox*, User's Guide, http://www.mathworks.com/access/helpdesk/help/toolbox/gads/gads.html.

Mathworks, 2004b, *Optimization Toolbox*, User's Guide, http://www.mathworks.com/access/helpdesk/help/toolbox/optim/.

Mathworks, 2004c, *SIMULINK: Dynamic System Simulation for MATLAB*, http://www.mathworks.com/access/helpdesk/help/toolbox/simulink/simulink.html.

Mattsson, S. E. and Andersson M., 1992, "Omola: An Object-Oriented Modelling Language", *Recent Advances in Computer-Aided Control Systems Engineering Studies in Automation and Control*, Edited by Jamshidi M. and Herget C. J., Elsevier Science Publishers, Amsterdam, Netherlands, pp. 291–310.

Max-Planck Institute of Computer Science, 2004, http://www.mpi-sb.mpg.de/LEDA/.

Michalewicz, Z., 1996, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, 3rd edition.

Michalewicz, Z., Dasgupta, D., Le Riche, R. G., and Schoenauer, M., 1996, "Evolutionary Algorithms for Constrained Engineering Problems", *Computers & Industrial Engineering Journal*, Vol.30, No.2, pp.851-870.

Miles, L., 1972, *Techniques of Value Analysis Engineering*, McGraw-Hill, New York, USA, http://www.wisc.edu/wendt/miles/milesbook.html.

MindTools, 2004, "Attribute Listing, Morphological Analysis & Matrix Analysis, Tools for creating new products & services", http://www.mindtools.com/pages/article/newCT_03.htm.

Mitchell, M., 1996, An *Introduction to Genetic Algorithms*, MIT Press, London, England.

Montana, D. J., 1995, "Strongly Typed Genetic Programming", *Evolutionary Computation*, 3-2.

Mosterman, P. J. and Biswas, G., 1995, "Behavior Generation using Model Switching: A Hybrid Bond Graph Modeling Technique", *Proceedings of the International Conference on Bond Graph Modeling'95*, vol. 27, no. 1, pp. 177-182, Society for Computer Simulation, San Diego, California, USA.

Otto, K. and Wood, K., 2001, *Product Design Techniques in Reverse Engineering and New Product Development*, Prentice Hall, New Jersey, USA.

Otto, K. and Wood, K., 1997, "Conceptual and Configuration Design of Products and Assemblies", *ASM Handbook, Materials Selection and Design*, Vol. 20, ASM International.

Özdemir, H. T. and Mohan, C. K., 2001, "Flight Graph Based Genetic Algorithm for Crew Scheduling in Airlines", *Information Sciences 133*, pp. 165-173.

Pahl, G. and Beitz, W., 1988, *Engineering Design: A Systematic Approach*, Springer-Verlag, UK.

Paredis, C. J. J., Diaz-Calderon A., Sinha R. and Khosla, P. K., 2001, "Composable Models for Simulation-Based Design", *Engineering with Computers 17*, pp. 112-128.

Piela, P., Epperly, T., Westerberg, K. and Westerberg, A., 1991, "ASCEND: An Object-Oriented Computer Environment for Modeling and Analysis: The Modeling Language", *Computers and Chemical Engineering*, Pergamon Press, Vol. 15, No. 1, pp.53-72.

Platin, B. E., Çalışkan, M., Özgüven, H. N., 1991, *Dynamics of Engineering Systems*, METU Press.

Pollack, J. B., Lipson. H., Ficici, S., Funes, P., Hornby, G. and Watson, R., 2000, "Evolutionary Techniques in Physical Robotics", *Proceedings of the third international conference on Evolvable Systems: from biology to hardware*, Edited by Miller, J., Springer (Lecture Notes in Computer Science; Vol. 1801), pp. 175-186.

Potter, S., Culley, S. J., Darlington J. M., Chawdhry, P. K., 2003, "Automatic Conceptual Design Using Experience-Derived Heuristics", *Research in Engineering Design*, Vol 14 No 3, pp. 131-144.

Rasheed, K., 1998, "An Adaptive Penalty Approach for Constrained Genetic-Algorithm Optimization", *Proceedings of 3rd Annual Genetic Programming Conference*, San Francisco, California, USA, pp. 584-590.

Reisig, W., 1992, *A Primer in Petri Net Design*, Springer-Verlag, Germany.

Reisig, W., 1985, *Petri Nets: An Introduction*, Springer-Verlag, Germany.

Ritchey, T., 2002, "General Morphological Analysis: A general method for non-quantified modeling", *An Article on Morphological Analysis*, Swedish Morphological Society, http://www.swemorph.com/ma.html.

Roston, G. P., 1994, *A Genetic Methodology for Configuration Design*, Ph.D. Dissertation, Carnegie Mellon University, Pittsburg, USA.

Rzevsky, G. 1995, *Designing Intelligent Machines*, Vol.1. The Open University, UK.

Sahlin, P., Bring, A. and Sowell, E. F., 1996, *The Neutral Model Format for Building Simulation, Version 3.02*, Technical Report, Department of Building Sciences, The Royal Institute of Technology, Stockholm, Sweden.

Salustri, F. A., 2000, *MEC 723 Mechanical System Lecture Notes*, http://deed.megan.ryerson.ca/~fil/L/Courses/mec723/2000/Lectures/lectures/node4.html.

Schaffer, J., 1985, "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms", *Proceedings of 1st International Conference on Genetic Algorithms*, Pittsburgh, USA.

Schulze-Kremer, S., 1996, *Genetic Algorithms and Protein Folding*, http://www.techfak.uni-bielefeld.de/bcd/Curric/ProtEn/contents.html.

Sembugamoorthy, V. and Chandrasekaran, 1986, "Functional Representation of Devices and Compilation of Diagnostic Problem-Solving Systems", *Experience, Memory and Reasoning*, Lawrence Erlbaum Associates, Hillsdale, New Jersey, USA.

Sharpe, J. E., 1995, "Conceptual Tools for Integrated Conceptual Design", *Design Studies Journal,* Vol: 16 (4), pp 471-488.

Sharpe, J. E. E. and Bracewell, R. H., 1995, "The Use of Functional Reasoning for the Conceptual Design of Interdisciplinary Schemes", *10th International Conference on Engineering Design*, ICED 95, Praha, Czech Republic.

Silva, S., 2004, GPLAB: A Genetic Programming Toolbox for MATLAB, http://gplab.sourceforge.net/.

Sims, K., 1994a, "Evolving Virtual Creatures", *Computer Graphics (Siggraph '94 Proceedings)*, pp. 15-22.

Sims, K., 1994b, "Evolving 3D Morphology and Behavior by Competition", *Artificial Life IV Proceedings, Edited by Brooks & Maes*, MIT Press, pp. 28-39.

Sinha, R., Liang, V. C., Paredis, C. J. J. and Khosla, P. K., 2001, "Modeling and Simulation Methods for Design of Engineering Systems", *Journal of Computing and Information Science in Engineering*, Vol. 1-1, pp. 84-91.

Smith A. E. and Coit D. W., 1997, "Constraint Handling Techniques-Penalty Functions", *Chapter C5.2 in Handbook of Evolutionary Computation*, Institute of Physics Publishing and Oxford University Press, Bristol, UK.

Smith, A. E. and Tate, D. M., 1993, "Genetic Optimization Using a Penalty Function", *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA-93)*, San Mateo, California, USA, pp. 499-503.

Snavely, G. L., Pomrehn, L. P. and Papalambros P. L., 1990, "Toward a Vocabulary for Classifying Research in Mechanical Design Automation", *Proceedings of the First International Workshop on Formal Methods in Design*, Colorado, USA, Jan 14-17.

Soemers, H. M. J. R., Cox, H., Gaal, E.W. and Eijk, van J., 2000, "The Mechatronic Design Approach, A Case Study", *Mechatronic 2000 Conference*, Atlanta, USA.

Stone, R., McAdams, D. and Kayyalethekkel, V., 2004, "A Product Architecture-Based Conceptual DFA Technique", *Design Studies*, 23(3):301-325.

Stone, R. B., Hirtz, J., McAdams, D. A., Szykman S. and Wood, K. L., 2001, "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts", *Journal of Research in Engineering Design,* 13(2), pp. 65-82.

Stone, R. and Wood, K., 2000, "Development of a Functional Basis for Design," *Journal of Mechanical Design*, 122(4):359-370.

Stone, R., Wood, K., and Crawford, R., 2000, "A Heuristic Method for Identifying Modules for Product Architectures", *Design Studies*, 21(1):5-31.

Stone, R. and Wood, K., 1999, "Development of a Functional Basis for Design", *Proceedings of DETC99, DETC99/DTM-8765*, Las Vegas, Nevada, USA.

Stone, R., Wood, K. and Crawford, R., 1998, "A Heuristic Method to Identify Modules from a Functional Description of a Product", *Proceedings of DETC98, DETC98/DTM-5642*, Atlanta, Georgia, USA.

Suh, N. P., 2001, *Axiomatic Design: Advances and Applications*, Oxford University Press, New York, USA.

Suh, N. P., 1998, "Axiomatic Design Theory for Systems", *Research in Engineering Design* (1998) 10, Springer-Verlag, London, UK, pp. 189-209.

Suh, N. P., 1990, *The Principles of Design*, Oxford University Press, New York, USA.

Summers, J. D. and Shah, J. J., 2003, "Developing Measures of Complexity for Engineering Design", *Proceedings of Design Engineering Technical Conference,* DETC'03, Chicago, Illinois, USA.

Sushil, J. L., 1997, Genetic Algorithms and Truss Design, http://www.cs.unr.edu/~sushil/papers/conference/papers/IJES/tech/node3.html.

Szykman, S., Racz J. W., Bochenek C. and Sriram R. D., 2000a, "A Web-based System for Design Artifact Modeling," *Design Studies*.

Szykman, S., Bochenek, C., Racz, J. W. and Sriram, R., 2000b, "Design Repositories: Next-Generation Engineering Design Databases", *IEEE Intelligent Systems and Their Applications*.

Szykman, S., Racz J. W. and Sriram R. D., 1999a, "The Representation of Function in Computer-based Design", *Proceedings of the 1999 ASME Design Engineering Technical Conferences (11th International Conference on Design Theory and Methodology)*, Paper No: DETC99/DTM-8742, Las Vegas, Nevada, USA.

Szykman, S., Sriram, R., Bochenek, C. and Racz J., 1999b, "The NIST Design Repository Project", *Advances in Soft Computing - Engineering Design and Manufacturing*, Roy, R., T. Furuhashi and P. K. Chawdhry (eds.), Springer-Verlag, London, UK, pp.5-19.

Terpenny, J. P., 1998, "Blending Top-Down and Bottom-Up Approaches in Conceptual Design", *7th Annual Industrial Engineering Research Conference*, Banff, Alberta, Canada.

Tummescheit, H., Klose, M., and Ernst, T., 1997, "Modelica and Smile: A Case Study Applying Object-Oriented Concepts to Multi-Facet Modeling", *Proceedings of the 9th European Simulation Symposium*, ESS97, Budapest, Hungary.

Ullman, D., 1997, *The Mechanical Design Process*, McGraw-Hill, New York, 2nd edition.

Ullman, D., G., 1993, "The Evolution of Function and Behavior during Mechanical Design", *Design Theory and Methodology ASME*, New York, USA, Volume 53, pp.91-103.

Ullman, D. G., 1992, *The Mechanical Design Process*, McGraw-Hill Inc., USA.

Ulrich, K. T. and Eppinger, S. D., 1999, *Product Design and Development*, McGraw-Hill, USA.

Ulrich, K. and Tung, K., 1991, "Fundamentals of Product Modularity", *Proceedings of the 1991 Winter Annual Meeting*, DE-Vol. 39 Atlanta, Georgia, USA, pp.73-79.

VAI (Value Analysis Incorporated), 1993, *Value Analysis, Value Engineering, and Value Management*, Clifton Park, New York, USA.

Ward, A. C. and Seering, W. P., 1989a, "The Performance of a Mechanical Design Compiler", *MIT Artificial Intelligence Laboratory AI Memo No. 1084*, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA.

Wall, 1996, *GAlib: A C++ Library of Genetic Algorithm Components*, User Manual, http://lancet.mit.edu/ga/.

Ward, A. C. and Seering, W. P., 1989b, "Quantitative Inference in a Mechanical Design Compiler", *Artificial Intelligence Laboratory AI Memo No. 1062*, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA.

Weber, R. G. and Condoor, S. S., 1998, "Conceptual Design Using a Synergistically Compatible Morphological Matrix", *Proceedings of Frontiers in Education Conference*, Arizona, USA.

Yoshikawa, H., 1985, "Design Theory for CAD/CAM Integration", *Annals of the CIRP*, Vol 34 No 1, pp.173–178.

Yoshikawa, H., 1981, "General Design Theory and a CAD system", *IFIP Man–Machine Communication in CAD/CAM*, Edited by Sata, T. and Warman, E., pp. 35–88.

Zhang, B. T. and Kim, J. J., 2000, "Comparison of Selection Methods for Evolutionary Optimization", *International Journal of Evolutionary Optimization*, Vol: 2-1, pp. 55-70.

Ziaei, R. and Agha, G., 2003, "SynchNet: A Petri Net Based Coordination Language for Distributed Objects", *Generative Programming and Component Engineering* (GPCE).

Zwicky, F., 1948, "The Morphological Method of Analysis and Construction", *Courant Anniversary Volume*, New York Wiley-Interscience, New York, USA.

APPENDIX A

RECONCILED FUNCTIONAL BASIS

A.1 FLOW DEFINITIONS (Stone et al., 2001)

A.1.1 Material

- Human. All or part of a person who crosses the device boundary. Example: Most coffee makers require the flow of a *human hand* to actuate (or start) the electricity and thus heat the water.
- Gas. Any collection of molecules characterized by random motion and the absence of bonds between the molecules. Example: An oscillating fan moves air by rotating blades. The air is transformed as *gas* flow.
- Liquid. A readily flowing fluid, specifically having its molecules moving freely with respect to each other, but because of cohesive forces, not expanding indefinitely. Example: The flow of water through a coffee maker is a *liquid flow*.
- Solid. Any object with mass having a definite, firm shape. Example: The flow of sandpaper into a hand sander is transformed into a *solid* entering the sander.
    - Object. Material that can be seen or touched that occupies space. Example: The box of scrap paper for recycling is represented as the flow *object*.
    - Particulate. Substance containing minute separate particles. Example: Granular sugar and powdered paint are *particulates*.
    - Composite. Solid material composed of two or more substances having different physical characteristics and in which each

substance retains its identity while contributing desirable properties to the whole unit. Any class of high-strength, lightweight engineering materials consisting of various combinations of alloys, plastics, and ceramics. Example: Materials such as wood, fiberglass combined with metals, ceramics, glasses, or polymers together are considered a *composite*.

- Plasma. A collection of charged particles that is electrically neutral exhibiting some properties of a gas, but differing from a gas in being a good conductor of electricity and in being affected by a magnetic field. Example: Plasma cutting focuses an intense beam of ionized air, known as *plasma*, produced by an electric arc, which melts the material to be cut.
- Mixture. A substance containing two or more components, which are not in fixed proportions, do not lose their individual characteristics and can be separated by physical means.
    - Liquid-Liquid. A readily flowing combination of two or more fluids, specifically having its molecules moving freely with respect to each other, but because of cohesive forces, not expanding indefinitely. Example: Machine oil and gasoline is a common *liquid-liquid* mixture used in yard maintenance machines.
    - Gas-Gas. A collection of molecules containing two or more components, which are characterized by random motion and the absence of bonds between the molecules. Example: The mixture of argon and carbon dioxide, a *gas-gas* flow, is commonly used in welding.
    - Solid-Solid. A combination of two or more objects with mass having definite, firm shape. Example: Pebbles, sand, gravel, and slag can be used to form concrete, mortar, or plaster. After it cures, concrete is a *solid-solid.*
    - Solid-Liquid. A combination of two or more components containing at least one solid and one liquid. Example: Iced tea

is a *solid-liquid* mixture of ice (solid), water (liquid), and tea grounds (solid).

- Solid-Gas. A combination of two or more components containing at least one solid and one gas. Example: Fog is a *solid-gas* mixture of frozen ice particles (solid) in air (gas).
- Liquid-Gas. A combination of two or more components containing at least one liquid and one gas. Example: Carbonated drinks are *liquid-gas* mixtures of flavored syrup (liquid), purified water (liquid), and carbon dioxide (gas).
- Solid-Liquid-Gas. A combination or three or more components containing at least one each of a solid, liquid, and gas. Example: In a cup of soda and ice cubes, the cup contains the *solid-liquid-gas* flow.
- Colloidal. A solid, liquid, or gaseous substance made up of very small, insoluble, non-diffusible particles that remain in suspension in a surrounding solid, liquid, or gaseous medium of a different matter. Example: Aerosols, smoke, and mist can all be considered *colloids*. Mist is a combination of very fine water droplets suspended in air.

A.1.2 Energy

Generic Complements:

Effort: Any component of energy used to accomplish an intended purpose.

Flow: Any component of energy causing the intended object to move or run freely.

- Human. Work performed by a person on a device. Example: An automobile requires the flow of *human energy* to steer and accelerate the vehicle.
    - Force. Human effort that is input to the system without regard for the required motion. Example: *Human force* is needed to actuate the trigger of a toy gun.

    - Velocity. Activity requiring movement of all or part of the body through a prescribed path. Example: The track pad on a laptop computer receives the flow of *human velocity* to control the cursor.
- Acoustic. Work performed in the production and transmission of sound. Example: The motor of a power drill generates the flow of *acoustic energy* in addition to the torque.
    - Pressure. The pressure field of the sound waves. Example: A condenser microphone has a diaphragm, which vibrates in response to *acoustic pressure*. This vibration changes the capacitance of the diaphragm, thus superimposing an alternating voltage on the direct voltage applied to the circuit.
    - Particle velocity. The speed at which sound waves travel through a conducting medium. Example: Sonar devices rely on the flow of *acoustic particle velocity* to determine the range of an object.
- Biological. Work produced by or connected with plants or animals. Example: In poultry houses, grain is fed to chickens, which is then converted into *biological energy*.
    - Pressure. The pressure field exerted by a compressed biological fluid. Example: The high concentration of sugars and salts inside a cell causes the entry, via osmosis, of water into the vacuole, which in turn expands the vacuole and generates a hydrostatic *biological pressure*, called turgor, that presses the cell membrane against the cell wall. Turgor is the cause of rigidity in living plant tissue.
    - Volumetric flow. The kinetic energy of molecules in a biological fluid flow. Example: Increased metabolic activity of tissues such as muscles or the intestine automatically induces increased *volumetric flow* of blood through the dilated vessels.
- Chemical. Work resulting from the reactions by which substances are produced from or converted into other substances. Example: A battery converts the flow of *chemical energy* into electrical energy.

- Affinity. The force with which atoms are held together in chemical bonds. Affinity is proportional to the chemical potential of a compound's constituent species. Example: An internal combustion engine transforms the chemical *affinity* of the gas into a mechanical force.
- Reaction rate. The speed or velocity at which chemical reactants produce products. Reaction rate is proportional to the mole rate of the constituent species. Example: Special coatings on automobile panels stop the *chemical reaction rate* of the metal with the environment.

- Electrical. Work resulting from the flow of electrons from a negative to a positive source. Example: A power belt sander imports a flow of *electrical energy* (electricity, for convenience) from a wall outlet and transforms it into a rotation.
  - Electromotive force. Potential difference across the positive and negative sources. Example: Household electrical receptacles provide a flow of *electromotive force* of approximately 110 V.
  - Current. The flow or rate of flow of electric charge in a conductor or medium between two points having a difference in potential. Example: Circuit breakers trip when the *current* exceeds a specified limit.
- Electromagnetic. Energy that is propagated through free space or through a material medium in the form of electromagnetic waves. It has both wave and particle-like properties. Example: Solar panels convert the flow *electromagnetic energy* into electricity.
  - Optical. Work associated with the nature and properties of light and vision. Also, a special case of solar energy (see solar). Example: A car visor refines the flow of *optical energy* that its passengers receive.
    - Intensity. The amount of optical energy per unit area. Example: Tinted windows reduce the *optical intensity* of the entering light.

        - Velocity. The speed of light in its conducting medium. Example: NASA developed and tested a trajectory control sensor (TCS) for the space shuttle to calculate the distance between the payload bay and a satellite. It relied on the constancy of the *optical velocity* flow to calculate distance from time of flight measurements of a reflected laser.
    - Solar. Work produced by or coming from the sun. Example: Solar panels collect the flow of *solar energy* and transform it into electricity.
        - Intensity. The amount of solar energy per unit area. Example: A cloudy day reduces the *solar intensity* available to solar panels for conversion to electricity.
        - Velocity. The speed of light in free space. Example: Unlike most energy flows, *solar velocity* is a well-known constant.
- Hydraulic. Work that results from the movement and force of a liquid, including hydrostatic forces. Example: Hydroelectric dams generate electricity by harnessing the *hydraulic energy* in the water that passes through the turbines.
    - Pressure. The pressure field exerted by a compressed liquid. Example: A hydraulic jack uses the flow *hydraulic pressure* to lift heavy objects.
    - Volumetric flow. The movement of fluid molecules. Example: A water meter measures the *volumetric flow* of water without a significant pressure drop in the line.
- Magnetic. Work resulting from materials that have the property of attracting other like materials, whether that quality is naturally occurring or electrically induced. Example: The *magnetic energy* of a magnetic lock is the flow that keeps it secured to the iron based structure.
    - Magnetomotive force. The driving force which sets up the magnetic flux inside of a core. Magnetomotive force is directly

proportional to the current in the coil surrounding the core. Example: In a magnetic door lock, a change in *magnetomotive force* (brought about by a change in electrical current) allows the lock to disengage and the door to open.

- Magnetic flux rate. Flux is the magnetic displacement variable in a core induced by the flow of current through a coil. The magnetic flow variable is the time rate of change of the flux. The voltage across a magnetic coil is directly proportional to the time rate of change of magnetic flux. Example: A magnetic relay is a transducer that senses the time rate of change of *magnetic flux* when the relay arm moves.

- Mechanical. Energy associated with the moving parts of a machine or the strain energy associated with a loading state of an object. Example: An elevator converts electrical or hydraulic energy into mechanical energy.
  - Rotational energy. Energy that results from a rotation or a virtual rotation. Example: Customers are primarily concerned with the flow of *rotational energy* from a power screwdriver.
    - Torque. Pertaining to the moment that produces or tends to produce rotation. Example: In a power screwdriver, electricity is converted into rotational energy. The more specific flow is *torque*, based on the primary customer need to insert screws easily, not quickly.
    - Angular velocity. Pertaining to the orientation or the magnitude of the time rate of change of angular position about a specified axis. Example: A centrifuge is used to separate out liquids of different densities from a mixture. The primary flow it produces is that of *angular velocity*, since the rate of rotation about an axis is the main concern.
  - Translational energy. Energy flow generated or required by a translation or a virtual translation.

        - Force. The action that produces or attempts to produce a translation. Example: In a tensile testing machine, the primary flow of interest is that of a *force* which produces a stress in the test specimen.
        - Linear velocity. Motion that can be described by three component directions. Example: An elevator car uses the flow of *linear velocity* to move between floors.
- Pneumatic. Work resulting from a compressed gas flow or pressure source.
    - Pressure. The pressure field exerted by a compressed gas. Example: Certain cylinders rely on the flow of *pneumatic pressure* to move a piston or support a force.
    - Mass flow. The kinetic energy of molecules in a gas flow. Example: The *mass flow* of air is the flow that transmits the thermal energy of a hair dryer to damp hair.
- Radioactive (Nuclear). Work resulting from or produced by particles or rays, such as alpha, beta and gamma rays, by the spontaneous disintegration of atomic nuclei. Example: Nuclear reactors produce a flow of *radioactive energy* which heats water into steam and then drives electricity generating turbines.
    - Intensity. The amount of radioactive particles per unit area. Example: Concrete is an effective radioactive shielding material, reducing the *radioactive intensity* in proportion to its thickness.
    - Decay rate. The rate of emission of radioactive particles from a substance. Example: The *decay rate* of carbon provides a method to date pre-historic objects.
- Thermal. A form of energy that is transferred between bodies as a result of their temperature difference. Example: A coffee maker converts the flow of electricity into the flow of *thermal energy*, which it transmits to the water
    - Temperature. The degree of heat of a body. Example: A coffee maker brings the *temperature* of the water to boiling in order to siphon the water from the holding tank to the filter basket.

- Heat rate. The time rate of change of heat energy of a body. Example: Fins on a motor casing increase the flow *heat rate* from the motor by conduction (through the fin), convection (to the air) and radiation (to the environment).

A.1.3 Signal

- Status. A condition of some system, as in information about the state of the system. Example: Automobiles often measure the engine water temperature and send a *status signal* to the driver via a temperature gage.
  - Auditory. A condition of some system as displayed by a sound. Example: Pilots receive an *auditory signal*, often the words "pull up," when their aircraft reaches a dangerously low altitude.
  - Olfactory. A condition of some system as related by the sense of smell or particulate count. Example: Carbon monoxide detectors receive an *olfactory signal* from the environment and monitor it for high levels of CO.
  - Tactile. A condition of some system as perceived by touch or direct contact. Example: A pager delivers a *tactile signal* to its user through vibration.
  - Taste. A condition of some dissolved substance as perceived by the sense of taste. Example: In an electric wok, the *taste signal* from the human chef is used to determine when to turn off the wok.
  - Visual. A condition of some system as displayed by some image. Example: A power screwdriver provides a *visual signal* of its direction through the display of arrows on the switch.
- Control. A command sent to an instrument or apparatus to regulate a mechanism.

- Analog. A control signal sent by direct, continuous, measurable, variable physical quantities. Example: Turning the volume knob on a radio sends an *analog signal* to increase or decrease the sound level.
- Discrete. A control signal sent by separate, distinct, unrelated or discontinuous quantities. Example: A computer sends *discrete signals* to the hard disk controller during read/write operations.

A.2 FUNCTION DEFINITIONS (Stone et al., 2001)

Note that certain functions are limited to operate on certain types of flows. This restriction is typically given in the function definition and applies to all functions at sub-levels of the given function.

- Branch. To cause a flow (material, energy, signal) to no longer be joined or mixed.
  - Separate. To isolate a flow (material, energy, signal) into distinct components. The separated components are distinct from the original flow, as well as each other. Example: A glass prism *separates* light into different wavelength components to produce a rainbow.
    - Divide. To split up a flow into parts or to classify distinct parts of a flow. Example: A vending machine divides the solid form of coins into appropriate denominations.
    - Extract. To draw, or forcibly pull out, a flow. Example: A vacuum cleaner *extracts* debris from the imported mixture and exports clean air to the environment.
    - Remove. To take away a part of a flow from its prefixed place. Example: A sander *removes* small pieces of the wood surface to smooth the wood.
  - Distribute. To cause a flow (material, energy, signal) to break up. The individual bits are similar to each other and the

undistributed flow. Example: An atomizer *distributes* (or sprays) hair-styling liquids over the head to hold the hair in the desired style.

- Channel. To cause a flow (material, energy, signal) to move from one location to another location.
    - Import. To bring in a flow (material, energy, signal) from outside the system boundary. Example: A physical opening at the top of a blender pitcher *imports* a solid (food) into the system. Also, a handle on the blender pitcher imports a human hand.
    - Export. To send a flow (material, energy, signal) outside the system boundary. Example: Pouring blended food out of a standard blender pitcher *exports* liquid from the system. The opening at the top of the blender is a solution to the export sub-function.
    - Transfer. To shift, or convey, a flow (material, energy, signal) from one place to another.
        - Transport. To move a material from one place to another. Example: A coffee maker *transports* liquid (water) from its reservoir through its heating chamber and then to the filter basket.
        - Transmit. To move an energy from one place to another. Example: In a hand held power sander, the housing of the sander *transmits* human force to the object being sanded.
    - Guide. To direct the course of a flow (material, energy, signal) along a specific path. Example: A domestic HVAC system *guides* gas (air) around the house to the correct locations via a set of ducts.
        - Translate. To fix the movement of a flow by a device into one linear direction. Example: In an assembly line, a conveyor belt *translates* partially completed products from one assembly station to another.

        - Rotate. To fix the movement of a flow by a device around one axis. Example: A computer disk drive *rotates* the magnetic disks around an axis so that the head can read data.
        - Allow degree of freedom (DOF). To control the movement of a flow by a force external to the device into one or more directions. Example: To provide easy trunk access and close appropriately, trunk lids need to move along a specific degree of freedom. A four bar linkage *allows a rotational DOF* for the trunk lid.
- Connect. To bring two or more flows (material, energy, signal) together.
    - Couple. To join or bring together flows (material, energy, and signal) such that the members are still distinguishable from each other. Example: A standard pencil couples an eraser and a writing shaft. The coupling is performed using a metal sleeve that is crimped to the eraser and the shaft.
        - Join. To couple flows together in a predetermined manner. Example: A ratchet *joins* a socket on its square shaft interface.
        - Link. To couple flows together by means of an intermediary flow. Example: A turnbuckle *links* two ends of a steering cable together.
    - Mix. To combine two flows (material, energy, and signal) into a single, uniform homogeneous mass. Example: A shaker *mixes* a paint base and its dyes to form a homogeneous liquid.
- Control Magnitude. To alter or govern the size or amplitude of a flow (material, energy, signal).
    - Actuate. To commence the flow of energy, signal, or material in response to an imported control signal. Example: A circuit switch *actuates* the flow of electrical energy and turns on a light bulb.

- Regulate. To adjust the flow of energy, signal, or material in response to a control signal, such as a characteristic of a flow. Example: Turning the valves *regulates* the flow rate of the liquid flowing from a faucet.
    - Increase. To enlarge a flow in response to a control signal. Example: Opening the valve of a faucet further *increases* the flow of water.
    - Decrease. To reduce a flow in response to a control signal. Example: Closing the valve further *decreases* the flow of propane to the gas grill.
- Change. To adjust the flow of energy, signal, or material in a predetermined and fixed manner. Example: In a hand held drill, a variable resistor *changes* the electrical energy flow to the motor thus changing the speed the drill turns.
    - Increment. To enlarge a flow in a predetermined and fixed manner. Example: A magnifying glass *increments* the visual signal (i.e. the print) from a paper document.
    - Decrement. To reduce a flow in a predetermined and fixed manner. Example: The gear train of a power screwdriver *decrements* the flow of rotational energy.
    - Shape. To mold or form a flow. Example: In the auto industry, large presses *shape* sheet metal into contoured surfaces that become fenders, hoods and trunks.
    - Condition. To render a flow appropriate for the desired use. Example: To prevent damage to electrical equipment, a surge protector *conditions* electrical energy by excluding spikes and noise (usually through capacitors) from the energy path.
- Stop. To cease, or prevent, the transfer of a flow (material, energy, signal). Example: A reflective coating on a window *stops* the transmission of UV radiation through a window.

        - Prevent. To keep a flow from happening. Example: A submerged gate on a dam wall *prevents* water from flowing to the other side.
        - Inhibit. To significantly restrain a flow, though a portion of the flow continues to be transferred. Example: The structures of space vehicles *inhibits* the flow of radiation to protect crew and cargo.
- Convert. To change from one form of a flow (material, energy, signal) to another. For completeness, any type of flow conversion is valid. In practice, conversions such as convert electricity to torque will be more common than convert solid to optical energy. Example: An electrical motor *converts* electricity to rotational energy.
- Provision. To accumulate or provide a material or energy flow.
    - Store. To accumulate a flow. Example: A DC electrical battery *stores* the energy in a flashlight.
        - Contain. To keep a flow within limits. Example: A vacuum bag *contains* debris vacuumed from a house.
        - Collect. To bring a flow together into one place. Example: Solar panels *collect* ultraviolet sun rays to power small mechanisms.
    - Supply. To provide a flow from storage. Example: In a flashlight, the battery *supplies* energy to the bulb.
- Signal. To provide information on a material, energy or signal flow as an output signal flow. The information providing flow passes through the function unchanged.
    - Sense. To perceive, or become aware, of a flow. Example: An audiocassette machine *senses* if the end of the tape has been reached.
        - Detect. To discover information about a flow. Example: A gauge on the top of a gas cylinder *detects* proper pressure ranges.

        - Measure. To determine the magnitude of a flow. Example: An analog thermostat *measures* temperature through a bimetallic strip.
    - Indicate. To make something known to the user about a flow. Example: A small window in the water container of a coffee maker *indicates* the level of water in the machine.
        - Track. To observe and record data from a flow. Example: By *tracking* the performance of batteries, the low efficiency point can be determined.
        - Display. To reveal something about a flow to the mind or eye. Example: The xyz-coordinate display on a vertical milling machine *displays* the precise location of the cutting tool.
    - Process. To submit information to a particular treatment or method having a set number of operations or steps. Example: A computer *processes* a login request signal before allowing a user access to its facilities.
- Support. To firmly fix a material into a defined location, or secure an energy or signal into a specific course.
    - Stabilize. To prevent a flow from changing course or location. Example: On a typical canister vacuum, the center of gravity is placed at a low elevation to *stabilize* the vacuum when it is pulled by the hose.
    - Secure. To firmly fix a flow path. Example: On a bicycling glove, a Velcro strap *secures* the human hand in the correct place.
    - Position. To place a flow (material, energy, signal) into a specific location or orientation. Example: The coin slot on a soda machine *positions* the coin to begin the coin evaluation and transportation procedure.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Güroğlu, Serkan

Nationality: Turkish (TC)

Date and Place of Birth: 24 April 1975 , Ankara

Marital Status: Single

Phone: +90312 586 8308

Fax: +90312 586 8091

email: serkan.guroglu@gmail.com

EDUCATION

Degree	Institution	Year of Graduation
M.Sc.	METU Mechanical Engineering	1999
B.Sc.	METU Mechanical Engineering	1997
High School	Ankara Kurtuluş High School	1992

ACADEMIC EXPERIENCE

Year	Place	Enrollment
2003-2005	Atılım University, Department of Mechatronics Engineering	Instructor
1998-2003	METU, Department of Mechanical Engineering	Research Assistant
1996	MKEK ELSA A.Ş.	Intern Engineering Student
1995	MKEK KAPSÜLSAN A.Ş.	Intern Engineering Student

FOREIGN LANGUAGES

Advanced English

PUBLICATIONS

1. Güroğlu, S., Erden, A., Akbulut, D. and Özgüç, B., 2005, "Creativity: As an Evaluation Measure for Evolutionary Conceptual Design", Submitted to the Sixth International Conference on Computer Aided Industrial Design and Conceptual Design, Delft/NETHERLANDS.
2. Güroğlu S. and Erden A., 2004, "Development of an Evaluation Measure for Genetic Approach to Functional Level Conceptual Mechatronic Design", 9th Mechatronics Forum - International Conference, MECHATRONICS 2004, Ankara/TURKEY.
3. Güroğlu S. and Erden A., 2004, "An Evolutionary Approach for Functional Level Conceptual Design of Products", International Design Conference, DESIGN 2004, Dubrovnik/CROTIA.
4. Güroğlu S. and Erden A., 2003, "Development of a Representation Scheme for the Application of Genetic Methodology to Functional Design", International Conference on Engineering Design, ICED 2003, Stockholm/SWEDEN.
5. Güroğlu S., Çetin O. and Erden A., 2001, "Application of a Pruning Algorithm to Automate the Design of Artificial Neural Networks", 5th International Conference on Mechatronic Design and Modeling, MDM 2001, Konya/TURKEY.
6. Güroğlu S., Erden Z. and Erden A., 2001, "Implementation of Petri Net Based Design Network on the Automation of Mechatronic Design", International Conference on Engineering Design, ICED 2001, Glasgow/UK.
7. Erden Z., Güroğlu S. and Erden A., 2000, "Functional Synergy in Mechatronic Design By Integrating Petri Nets and Hybrid Automata", 7th Mechatronics Forum International Conference, Mechatronics 2000, Atlanta, Georgia/USA.
8. Güroğlu S., Erden Z., Erkmen A., and Erden A., 1999, "The Design Network Simulator (DNS): An Implementation Software for a Petri Net-Based Design Network Applied to Mechatronic Design", 6th International Conference on Mechatronics and Machine Vision in Practice, M2ViP'99, Ankara/TURKEY.