2007

A virtual engineering framework to support progressive interaction in engineering design

Balasubramaniam Karthikeyan
Iowa State University

A virtual engineering framework to support progressive interaction in engineering design

by

Balasubramaniam Karthikeyan

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Mechanical Engineering

Iowa State University

Ames, Iowa,

2007

Dedication

To Guru and Krsna

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my heartfelt gratitude and appreciation to my major professor, Dr. Kenneth “Mark” Bryden, for his encouragement, guidance and support throughout my doctoral studies. I am extremely grateful to Dr. Daniel A. Ashlock for his valuable inputs in the inter-disciplinary areas like Computational Geometry, and Evolutionary Optimization techniques. The encouragement and valuable feedback from Dr. Carolyn Heising, Dr. Eliot H. Winer and Dr. Ron M. Nelson are sincerely acknowledged. I owe my thanks to Dr. Judy M. Vance who formally introduced me to the world of Computer Graphics and Geometric Modeling that forms the basis of this research work. The excellent group dynamics and support provided by my colleagues from Dr. Bryden research group deserve special appreciation. I am indebted to all my friends and well-wishers for their moral support while I went through ups and downs during my doctoral studies. The significant role played by my wife in assisting me in my PhD process is duly recognized. Last, but not the least, I would like to thank my parents for their patience, prayers, and complete confidence in me while I pursued my doctoral studies.

Abstract

Engineering design encompasses a series of non-trivial decision making phases in generating initial solutions, developing mathematical models, performing analysis, and optimizing designs. Engineering analysis and optimization are the phases that often significantly slow down the design process. Thorough designer exploration on the solution space increases the likelihood of determining the most feasible solution but, at the expense of longer lead times. The exploratory capabilities of the designer could be enhanced by creating an interactive virtual engineering framework. This research presents progressive interaction with the designer-in-the-loop whose intelligence is blended with the computational power to suitably control the optimization. Progressive interaction is a human-guided preference articulation method where the designer intelligence continuously controls the engineering analysis and optimization by visualization, modification and controlled re-optimization. Based on the designer's knowledge and the knowledge available from the interaction system, the designer preferences can be modified anytime to expedite optimization. Progressive interaction not only helps the designer discover the hidden relationship between the decision variables but it also uncovers the implicit constraints and other performance limitations of the design. In summary, this research work proposes human-guided, progressive interaction as a solution to complex engineering optimization problems. The proposed solution is demonstrated using three test cases: 1. Interactive image segmentation and optimization, 2. Designer interaction to support shape optimization of a finned dissipater, and 3. Interactive analysis, optimization and design of hydraulic mixing nozzle.

CHAPTER 1. INTRODUCTION

The decision making process, in general, can be split into four main phases: intelligence, design, choice, and review [Simon, 1960]. Identification of the decision problem and collection of relevant information is done in the intelligence phase. After initial screening, the design phase generates smaller number of decision alternatives for further consideration. In the choice phase, detailed analysis is performed on the solutions that were short-listed from the design phase. The solution chosen is implemented in the review phase [Lotov et al., 2004]. The set of tasks involved within each phase of the decision making process are problem and domain specific. Decision-making encompasses a wide range of application areas including business, economics, science, engineering, medicine, and geographical information systems. This research focuses on the decision- making issues pertinent to engineering. Within engineering decision-making is applicable to a wide variety of areas, including product design, analysis, manufacturing, logistics, marketing, sales. Each area requires specialized research to completely address their unique decision making issues. This research focuses on the decision-making issues within engineering design. Engineering design has been chosen for this research because it forms the fundamental building block of product development process. In addition, the critical decisions made during the engineering design phase have their impact on the product throughout its life cycle.

1.1 Elements of Engineering Design

Engineering design and decision-making are so intertwined that it has been suggested that all engineering design problems can be considered as decision making problems [Simon, 1969]. In addition, the terms “engineering decision-making” and “engineering design” are used interchangeably in most engineering design literature [Mistree and Allen, 1997]. Engineering design is a complex, non-trivial, and usually an iterative process where the knowledge available to the designer increases with time. Since the early 1960s many versions of engineering design tasks have been prescribed. Some researchers prescribed only four tasks, whereas others have decomposed engineering design into dozens of sub-tasks [Voland, 1999]. In this research work, the tasks considered for engineering design are problem definition, initial solution development, modeling, analysis and optimization. Engineering analysis and optimization are the tasks that are often the most time consuming and can significantly slow down the engineering design process. This research work is aimed at the development of interaction methods to facilitate engineering analysis and optimization.

An overall understanding of the engineering design process, in general, and in-depth understanding of the analysis and optimization tasks, helps in providing an appropriate engineering solution. The rest of this section is dedicated for the brief discussion on engineering design tasks.

Problem definition. The need for engineering design mostly arises due to one or more of the following, viz., to address the health, safety, or quality concerns of the public, to eliminate shortcomings in the original design or fabrication by incorporating new technology and manufacturing methods, to keep pace with the competition, and to reduce costs [Voland, 1999]. Once the need for engineering design is clearly identified, the designer is involved in problem definition. Problem definition includes the listing of customer specifications, the establishment of target product specifications, the preliminary analysis on the competitors’ product(s), the design issues with the existing product(s), and the list of constraints and tradeoffs. The quality of the engineering design depends on the clarity of problem definition. Hence, this task is critical. More details regarding this phase of engineering design can be accessed from [Ertas and Jones, 1996; Dieter, 2000].

Initial solution development. Developing good initial solutions is critical as the final design depends on the appropriateness of the initial solutions. Based on the designers' thinking perspective, the engineering design is subdivided into converging and diverging thinking. During the initial solution development, diverging thinking is applied to explore a variety of feasible design alternatives. The goal of the designer is to generate a number of candidate solutions which might not be optimal [Walker et al., 1991]. Initial solution development is a non-trivial task as the designers often have to work with incomplete information on an ill-defined problem, especially when a new product is developed. Past experience or *a priori* knowledge in the design of similar products facilitates the development of appropriate initial solutions.

Modeling. A model is an idealization or simplified representation of a system that is being developed. The models aid in the analysis of the design problems, for example control volume in a thermodynamic system is a simplified representation of the system. Designers generate a variety of models to represent the problem, to acquire knowledge, and to explore alternative solutions. Symbolic models describe the design using words, numbers, or mathematical equations [Birmingham et al., 1997]. Designers create the most common among the models, the mathematical models, with a hope to simulate the real physical system accurately. Mathematical modeling is the starting place for engineering optimization. Mathematical models help the designers and the analysts to understand the objective functions and the constraints present in the problem. In the initial stages of design, due to incomplete knowledge on the product the designers use their *a priori* knowledge to develop simplified mathematical models using assumptions. As the design process proceeds, the designer knowledge increases resulting the development of more accurate models.

Analysis. Engineering solutions undergo changes from one design generation to the next. Even within the same generation the solutions undergo major changes while moving along the engineering design process. Analysis is the stage where the performance of the designed product or system is verified against the design objectives using experimental or computational methods. The experimental methods of analysis are not preferred due to the cost, the time needed for development, and the inability to accept changes. Hence, computer-based analytical tools are preferred by the designer and the analysts. To ensure the acceptable performance of the designed product various analytical tools are used at different stages of product development. The most commonly used engineering tools include: computer aided design (CAD), finite element analysis (FEA), computational fluid dynamics (CFD), and rapid prototyping. In many cases these tools and others will be combined together to create an interactive virtual engineering (VE) environment. Virtual engineering is a designer-centered process that integrates the product models and engineering tools to facilitate interactive engineering design. These tools not only facilitate the engineering design process but also predict the product or system performance [Xiao and Bryden, 2004].

Optimization. This is an open-ended, iterative, and non-trivial process that involves incorporating multiple constraints and conflicting objectives in pursuit of a set of "best" solutions. Designers may change the design variables, constraints, or even design requirements. The optimization process will yield multiple feasible solutions. As a final step in the design process the engineer analyzes the efficacy of the obtained feasible solutions against the perceived goals of the needed design. If the design goals are met by the current optimum solution then the optimization is stopped, if not, the next iteration in the pursuit of identifying an optimum solution begins.

When a complex system is designed, the engineering analysis and optimization are the two tasks that consumes significant portion of design lead time. Despite such long exploration the “best” solutions are not always determined. For the want of time the engineers may opt for limited search space exploration. This compromise may lead to undesirable results due to the acceptance of sub-optimal solutions. This research focuses on studying the influence of designer interaction on engineering optimization by exploring fruitful regions of the search space. The most common interaction methods are reviewed as a prerequisite to develop designer-centered interaction.

1.2 Interaction Methods

The three most common methods of designers’ (or analysts’) preferences articulation within the optimization process are *a priori*, *posteriori*, or progressive [Van Veldhuizen and Lamont, 2000]. *A priori* and *posteriori* interaction are unguided or computer controlled search processes, as the designers have almost no control on the solutions returned by the optimization algorithms other than specifying their preferences before and after the optimization processes respectively. In most cases, the designer accepts feasible solutions presented by the optimization algorithms without a thorough exploration (“what-if” analysis) of various interesting regions of the search space. In contrast, progressive interaction is a guided or designer controlled search, as the designers’ preferences continuously direct the computational process. The preference articulation methods are presented in the following sub-sections.

1.2.1 A priori interaction

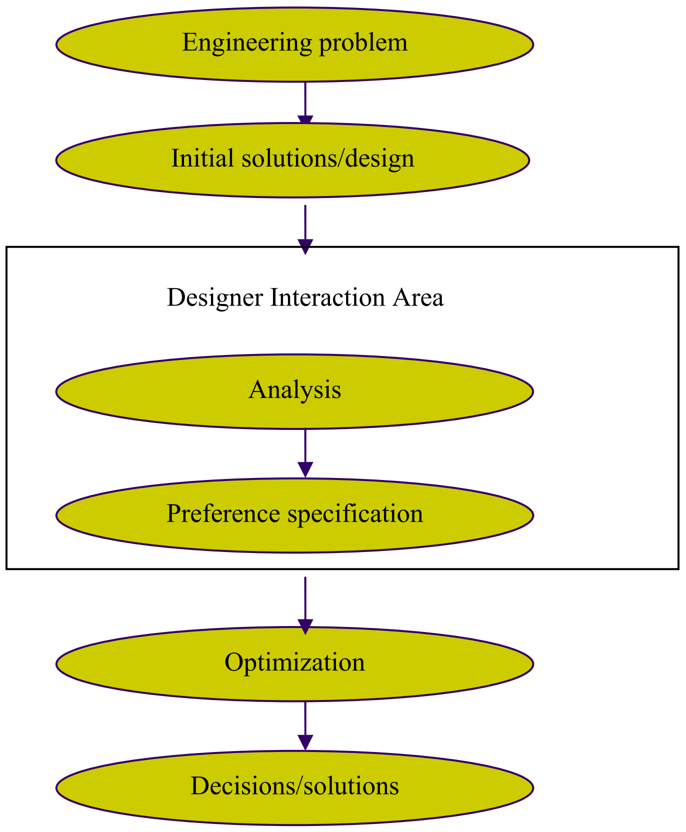


Fig 1.1 *A Priori* Preference Articulation

In *a priori* interaction the designer explicitly prescribes their preferences before the optimization process [Branke et al., 2001], as shown in Fig 1.1. The prescribed preferences narrow down the solution search space and hence, expedite the optimization process *en route* to engineering design. For example, a multi-objective optimization problem may be converted to a single-objective problem based on the designers' preferences and intuition without exploring possible alternatives. This interaction type is preferred when the designer is interested in rapidly determining the preliminary relationship between the decision variables and the objective function(s). The simplicity and the ease of implementation are the advantages of *a priori* interaction. The disadvantage of this interaction is its impracticability for the design of new, large, and complex systems. The quality of the design directly depends on the accuracy of the mathematical models.

1.2.2 Posteriori Interaction

In *posteriori* interaction designers articulate their preference *after* the complete results from the optimization are available [Branke et al., 2001], as shown in Fig. 1.2. The designer preferences are provided to the system based on thorough study of the available alternatives. This approach is well-suited for optimization problems where the designers have a reasonable (if not an exact) estimate of the expected solution and the relationship between the decision variables, objective functions, etc. In essence, the machine generates a palette of alternatives and the designers select their design from the alternatives. The advantages of *posteriori* interaction techniques are its simplicity of implementation, and its applicability over a wide range of optimization problems. The disadvantage of this interaction technique is felt when multiple iterative runs are to be conducted. The designers have almost no control on the search during the optimization process. Redefining the problem, constraints, and solution requirements can be done only when the current optimization run is completed or terminated.

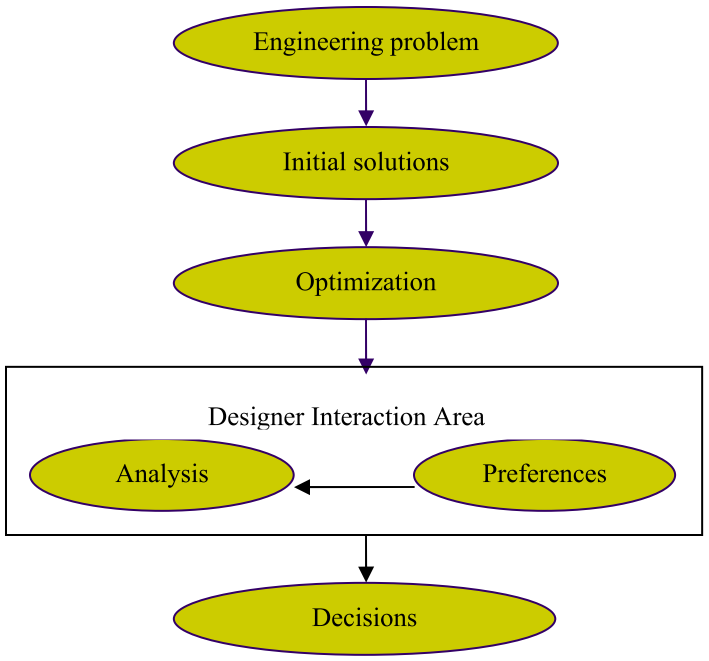


Fig 1.2 *Posteriori* Preference Articulation

1.2.3 Progressive Interaction

Progressive interaction is defined as a human-guided preference articulation method where the designers' interaction continuously controls the engineering optimization by visualization, modification and controlled re-optimization. In progressive interaction designer preference articulation is done continuously *during* the optimization process [Branke et al., 2001], as portrayed in Fig 1.3.

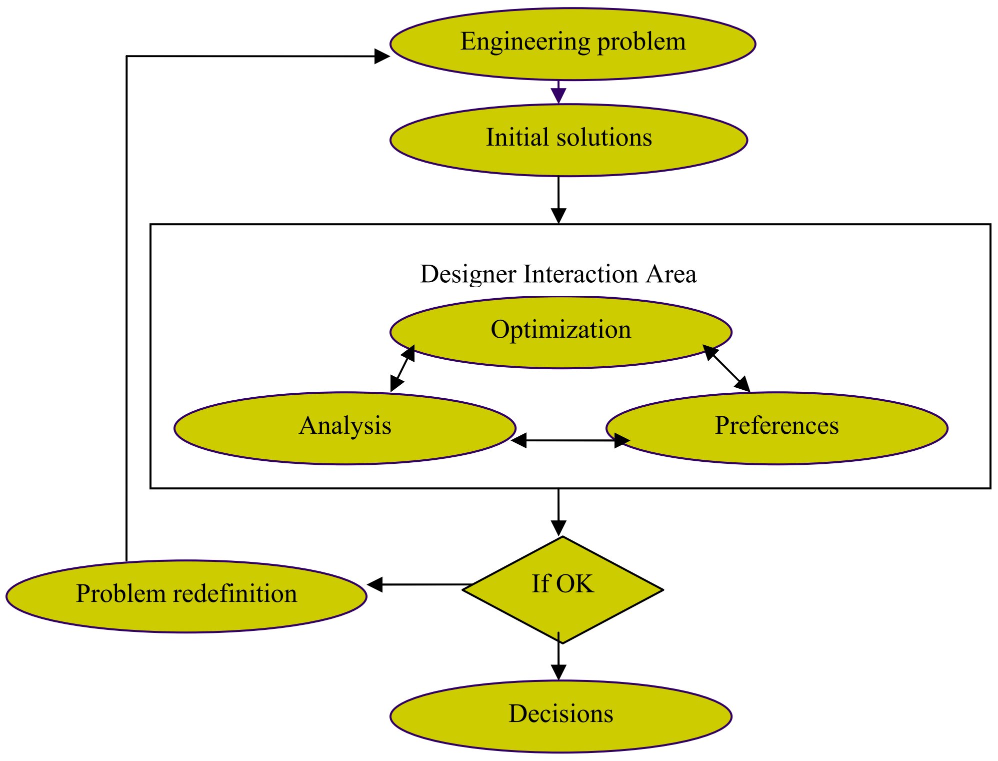


Fig 1.3 Progressive Preference Articulation

Based on the current results, the designer can modify their preferences periodically. The interesting area of the search space that may have been ignored can be explored using this interaction technique. The advantage of progressive interaction is its capability to handle engineering optimization involving complex, high fidelity analysis models, such as CFD, FEA, etc. When the complex analyses models are integrated within the engineering optimization process the time taken to perform even one optimization run is prohibitively long. In such a scenario, the progressive designer interaction controls the optimization algorithm by restricting the search only in the fruitful areas. The major disadvantage of progressive interaction technique is its implementation complexity.

1.3 Problem Specific Interaction

Engineering analysis and optimization are often the two processes that consume the most significant portion of product development lead-time. This is even truer when a large system consisting of sub-systems is to be developed. The larger the system, the higher the number of decision (or design) variables and hence the greater the complexity. The time for optimization increases with the problem complexity. The problem complexity arises due to the high-dimensionality i.e., large number of decision variables, the non-linearity of objective functions and constraints, and the difficulty to develop mathematical models. The problems can be classified *complex* according to the time it takes for an algorithm to solve them. In principle, there are problems that may be computationally solvable, but in reality, such problems require large amounts of time and space resources to solve. This introduces the following questions, 1. can the complex optimization problem be expedited by designer interaction? 2. and if interactive techniques can expedite optimization then what is the best interaction method?

At present, there is no one specific interaction technique applicable to all problem types. The final choice of an appropriate interaction scheme is problem specific. Based on the understanding from the current interaction techniques the designer interaction map, shown in Fig. 1.4, is proposed in this research work.

The Zone 1, of Fig. 1.4 indicates less complex problems with fewer objective functions and decision variables. Due to the lower level of complexity in the problem, the designers can develop the mathematical models using their *a priori* knowledge. The designer is aware of the relationship between the objective functions and their decision variables. Using this knowledge the designer can develop a composite objective function that takes into account all the objective functions based on their weights. *A priori* interaction will suffice for these problems.

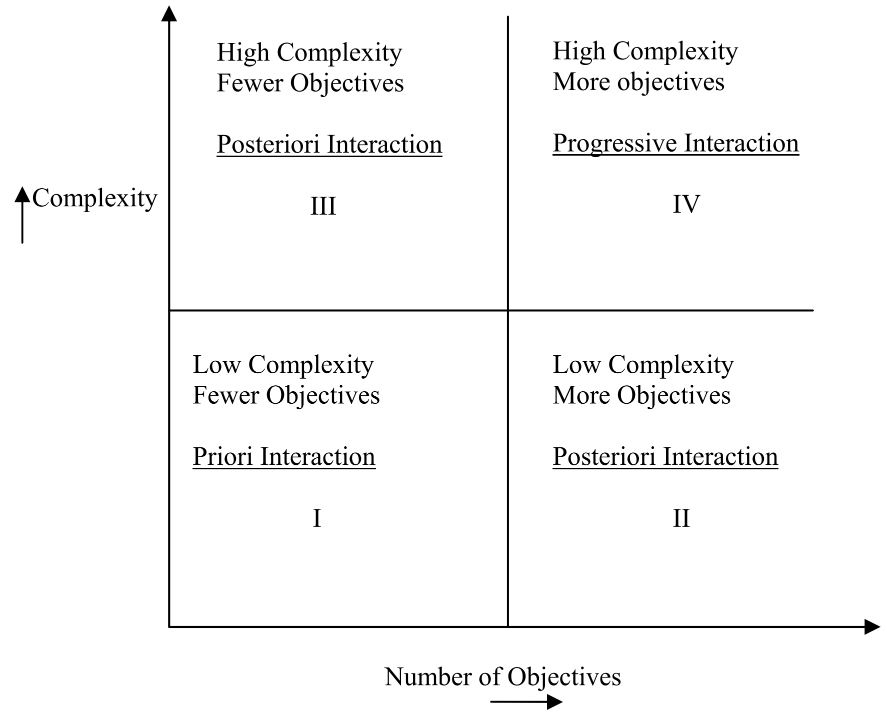


Fig 1.4 Designer Interaction Map

In Zone II, although the problems have a larger number of objective functions, due to the simplicity of the objective functions and constraints the problems are less complex. In this case the relationship between the decision variables and the decision variables are simple and *posteriori* designer interaction suffices.

Zone III deals with the problems those have fewer objective functions when compared to those in Zone II. Due to the complex relationships between the large number of decision variables, and objective functions, the problems are time consuming to solve. With a few computational runs the designer can gain an understanding of the nature of the problem, the interesting areas of the search space, etc. For the problems in Zone II and III, the *posteriori* interaction approach is preferred. The designer has the flexibility to delay the preference articulation until the complete results from the optimization are available.

In Zone IV, the problems have a large number of conflicting objective functions. Due to complex relationship between the objective functions, and constraints and the unknown relationships between decision variables these optimization problems become extremely challenging to solve. Understanding the implicit constraints and relationship from the experimental optimization runs may not be feasible, as the computational runs might take days, weeks, or months. For such complex problems, the exploration should be restricted to fruitful areas of the search space using designer intelligence. During such exploration, objective functions may need to be added, constraints may need to be hardened or softened to complete the optimization in a timely fashion. All of these are accomplished with a continuous interaction between the designer and computational tools. Hence, *progressive* or continuous designer interaction is required for this problem. Also the optimized solutions obtained in this zone are usually starting place for further analysis due to the multiple intangible constraints. The preliminary understanding of the optimization problem, and the relationships between the decision variables and the non-linear objective functions are studied simultaneously. Based on the understanding from the preliminary studies, the designers' choice of design variables, objective functions, and constraints are introduced to the optimization system. This may result in newer and better solutions as the optimization proceeds. To expedite engineering optimization of these problems greater level of designer interaction with the optimization system is desired when compared to those in the other three zones.

1.4 Motivation

The motivation for this research is to explore whether the optimization process can be expedited by designer interaction. The two prime goals that would be achieved include expedited engineering optimization and quicker, faster understanding of the design problem. Due to the available of specific knowledge the designers are superior to computers in areas like abstract thinking, analysis, and pattern recognition. Based on their *a priori* knowledge the designers could control the engineering analysis and optimization tools to operate in fruitful areas of the design space. Optimization techniques using high-fidelity models, such as CFD, and FEA, are extensively used in the engineering design today. However, often these optimization tools are implemented in a way that the designer interaction with the system is almost negligible *during* the optimization. Due to the lack of a structured interaction process, the designers' intelligence is not optimally blended with the computational power. Due to the limited computational assistance the designer intelligence is neither used to expedite the optimization nor to investigate the interesting regions of the search space. As a consequence the time spent on engineering optimization is often longer than needed and the best available solution may not be obtained. This situation is particularly true when large, complex engineering systems, for instance, automotive systems, are designed with multiple conflicting objectives/disciplines.

Evolutionary algorithms (EAs) are most commonly used in a variety of engineering optimization problems. EAs are preferred when the wide search space exploration is desired. These strategies rely upon a number of stochastic operators that maintain a high degree of exploration resulting in a broad sampling of available solutions. EAs become time-consuming when the engineering optimization involves high fidelity models such as CFD or FEA. Parmee [2001] proposed the integration of evolutionary and adaptive search optimization techniques with other complimentary computational intelligence techniques to enhance optimization. Despite the availability of a variety of computationally efficient optimization algorithms the time for engineering optimization involving high fidelity models are significantly high. The optimization time can be significantly reduced by reducing the number of calls to the computational solver of the high fidelity analysis or by using models of reduced fidelity. Using the models of reduced fidelity is inappropriate when the optimization involves complex thermo-fluid systems. Performing thousands of iterations in an engineering optimization is not uncommon. The number of calls to the computational solver is equal to the number of iterations. The calls to the computational solver can be significantly reduced when the designer intelligence is used to direct the optimization search in the fruitful areas of the solution space. This can be accomplished only when the designer is enabled to provide their preferences during optimization. This is the motivation for this research work. The research question that is answered by this work is: How can design optimization using high fidelity models, such as CAD, FEA, and CFD models, be accomplished as quickly and effectively as possible.

Interactive optimization is facilitated when the designer-in-the-loop is enabled to articulate their preferences *during* the optimization, to visualize the current results, to control the input parameters based on the results, and to direct the computations to fruitful areas of the search space. As discussed in Section 1.2, there are many different methods of developing an interactive optimization tools. Of the three types of designer interaction, progressive interaction is preferred because the impact of the variables (or preferences) provided by the designer can be studied without disturbing the optimization runs. Progressive interaction expedites the designer understanding of the product and which is particularly necessary to expedite engineering optimization. The prime goal of this research work is to integrate the evolutionary optimization techniques, high-fidelity engineering analysis models, and designer interaction in an immersive virtual engineering environment. The efficacy of the proposed interaction scheme is demonstrated using the following test cases:

1. *Interactive image segmentation and optimization*

   Evolutionary optimization is controlled by designer interaction to make engineering decisions using high-dimensional datasets, such as, digital images, within a virtual engineering environment.

2. *Designer interaction to support shape optimization of a finned heat exchanger*

   Computational fluid dynamics simulations and evolutionary optimization algorithms are integrated together in a virtual engineering environment that is controlled by the designer to facilitate the shape optimization of a thermo-fluid system.

3. *Interactive analysis, optimization, and design of hydraulic mixing nozzle*

   Evolutionary optimization technique is coupled with a commercial high-fidelity CFD package (Star-CD) to generate flow field simulations for all the candidate solutions.

CHAPTER 2. BACKGROUND AND LITERATURE REVIEW

The scope of the term "design" has been significantly expanded since 1960s. The meaning of "engineering design" for engineering students is promulgated by the accreditation board for engineering and technology (ABET). ABET defines, "Engineering design is the process of devising a system, component, or process to meet desired needs. It is a decision making process (often iterative), in which the basic science, mathematics, and engineering sciences are applied to convert resources optimally to meet a stated objective. Among the fundamental elements of the design process are the establishment of objectives and criteria, synthesis, analysis, construction, testing, and evaluation" [ABET, 1995].

Prior to 1960s, the term "design" referred to making drawings on a drafting table using drafting tools. Most young engineering graduates of 1960s that looked for challenging work preferred analysis over design [Ertas and Jones, 1996]. The ability of the designer to draw the product before the manufacturer built it was considered the original contribution to the design process. At that time, the designers' first concern was on the geometry that defined the shape of the product. In early 1960s, the industry and the academia from the US and Europe recognized the necessity to manage the design process and the overall product development process [Birmingham et al., 1997]. As a consequence, the engineering design became the prime area of research interest. The scope, approach, and methodology of engineering design have significantly changed over the past four decades. Even the role of the designer has changed from being a support person to being a lead person in the product development process. Section 2.1 discusses the prime research directions of the past that has resulted in the present days' design process and methodology. Section 2.2 presents the engineering design process in the light of section 2.1. Section 2.3 to 2.5 describes the role of computers in engineering design followed by the development of advanced technologies in virtual engineering. Sections 2.6 to 2.8 present engineering optimization techniques as an inseparable part of engineering design.

2.1 Evolution of Engineering Design

Over the last four decades the engineering design has evolved in a variety of ways. In this section, the five major areas chosen for the study of evolution of engineering design are: engineering design methodology, diffusion of concepts from other disciplines, integration of mathematical and scientific laws, design for X, and the role of designer. These areas are considered significant as they set the stage for the present days' research in engineering design.

2.1.1 Engineering design methodology

Harold Buhl [1960] in his book on creative engineering design proposed a series of steps applicable for most engineering design problems. The steps include problem identification, problem definition, solution development, analysis, synthesis, evaluation, and presentation. Design research conducted in early 1960s worked to identify and develop solutions processes pertinent to each step in engineering. Such research work resulted in overall management of engineering design process [Jones and Thornley, 1963]. Asimov [1962] was one of the first researchers to propose the basic modules of engineering design. Asimov also presented a detailed description of the complete design process which he called the *morphology of design*. Asimov described the morphology of design in seven phases: conceptual design, embodiment design, detailed design, planning for manufacture, planning for distribution, planning for use, planning for retirement of the product. The scope of our research work is to expedite the engineering design process. Hence, only the first three phases are considered in detail in future sections. Fig. 2.1 shows the basic modules in the design process proposed by Asimov.

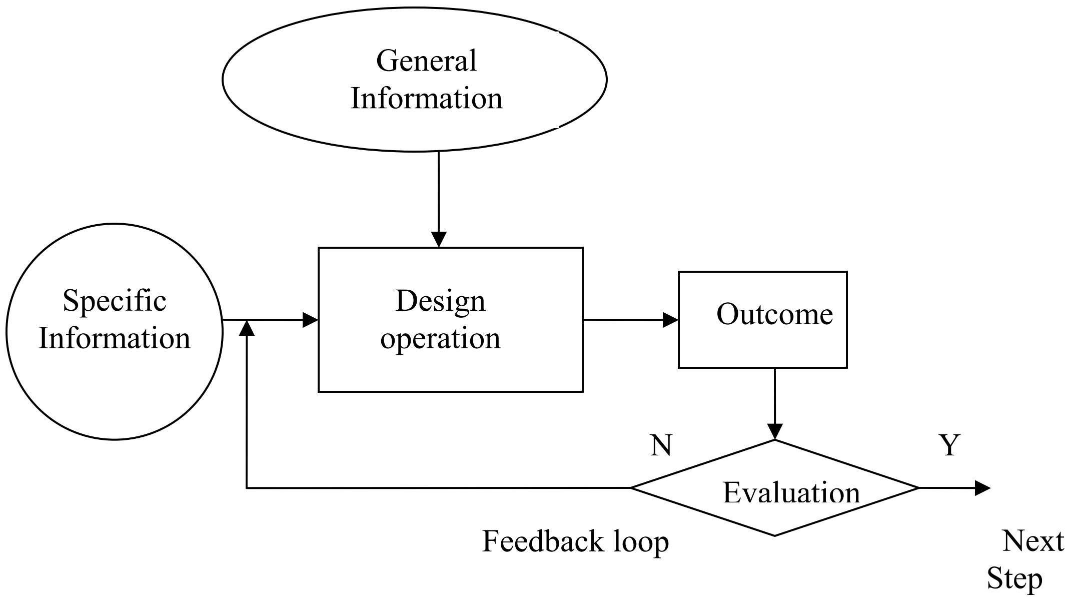


Fig. 2.1 Basic Module in the Design Process (Asimov, 1962)

2.1.2 Diffusion of concepts into engineering design

The research on engineering design also borrowed ideas from other disciplines that were matured. This was evident from the work of Hill [1970]. Hill compared the scientific method and the design method. A shown in Fig. 2.2 the scientific method starts with a body of existing knowledge. A hypothesis is formulated due to their scientific curiosity or interest to understand certain theory. The hypothesis is subjected to logical analysis that either confirms or denies it. Due to the flaws or inconsistencies often the hypothesis may have to be changed in an iterative process. The next steps in the scientific method are self-explanatory. Design method is similar to the scientific method as it starts with the knowledge of the state-of-the-art. This includes knowledge on devices, components, materials, manufacturing methods, market and economic conditions. Scientific curiosity is replaced by the identification of the needs of the society in the design method. The needs are conceptualized as some kind of model. The design concept is subjected to an iterative feasibility analysis until an acceptable product is developed.

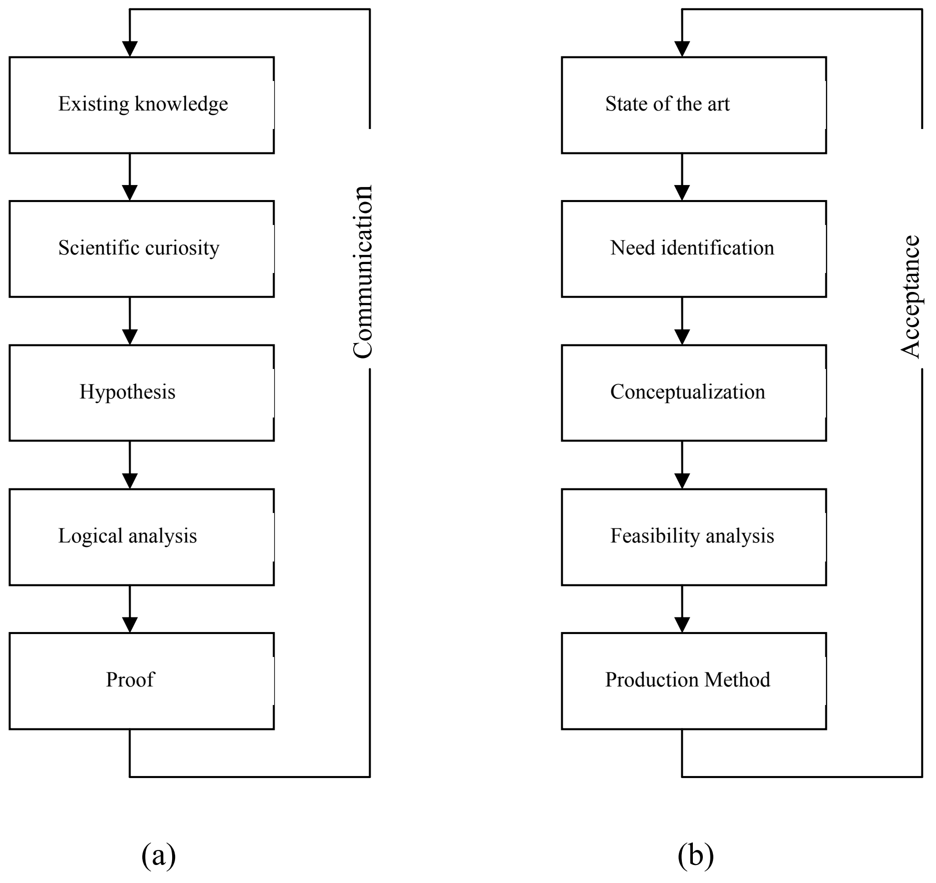


Fig. 2.2 Comparison between the scientific method and the design method (Hill, 1970)

2.1.3 Integration of mathematics and scientific laws

Cain [1969] classified the design projects based on the product size, technological content, complexity, and skill requirements of the designer. Based on the type of the design projects, the design personnel and their responsibilities varied. The projects that are technologically sophisticated and complex necessitated the determination of best design solution before they were implemented. This opened up the research in the area of engineering optimization that integrated engineering design and numerical optimization. Researchers understood the importance of integrating scientific laws and mathematical equations. The researchers changed the emphasis in engineering design from engineering design as an art to engineering design as a scientific problem solving technique. The vast majority of literature indicates that the researchers started focusing on developing numerical techniques using computers in early 1970s. Thus the mathematical theory of optimization became highly developed and was applied to design problems. Simon [1969] concentrated on developing a pool of feasible solutions from where the designer can analyze and determine the best solution. This technique is comparable to present days' optimization process. Rittel and Webber [1973] proposed two types of design problems: ill-structured and well-structured. Well-structured problems could be described exclusively in terms of numerical values such that the optimal solutions can be found using available algorithms. Ill-structured problems may have many possible solutions the "best" solutions to ill-defined problems depend on the designer priorities. In addition, the ill-structured problems have unclear goals and incomplete information. Many engineering design problems were classified under ill-structured problems. The research work in the field of solving ill-structured problems set the stage for the development of present days' optimization methods. Simon [1984] suggested the technique of thorough analysis and problem formulation to convert the ill-structured problems to well-structured problems.

2.1.4 Design for X

Cain [1969] had presented the concepts of designing for function, use, production and appearance. The present days' concepts of design for manufacture and assembly (DFM/DFA) have their roots in the history but they were documented as early as 1960s. The research interests on DFM/DFA didn't intensify until 1980s. Starting in the 1980s, the new approaches for the integrated product design evolved, concurrent engineering. Concurrent engineering is a systematic approach that focuses on simultaneous development of all aspects of product development from the initial design to its manufacture, maintenance, and disposal. Concurrent engineering eliminated the notion of sequential design process. Many professionals representing one or more areas of product development work together as a team throughout the design process to ensure the finest product quality in shortest lead time [Dieter, 2000]. This goal was achieved by avoiding unforeseen obstacles, if any, that would prevent the fructification of the engineering design. This stemmed the research towards the concept of design for X. The most common among the design for X include: design for manufacturing (DFM), and design for assembly (DFA).

2.1.5 Engineer – an innovative designer

The economic, environmental, and political factors necessitated the development of new technologies in engineering design and manufacturing in 1970s and 1980s. This motivated the then researchers to consider the design as an innovative process. Holt [1983] explained innovation in design as "a process which covers the use of knowledge or relevant information for the creation and introduction of something that is new and useful". Based on the level of innovation, the design activity was classified as original design, adaptive design, and variant design [Pahl and Beitz, 1984]. The adaptive and variant designs are preferred as they involve minimal risks when compared to the original design. The further research in innovation was directed towards the innovation strategies by Cooper [1984]. Birmingham et al., [1997] considered engineering design as innovative process that can be a source of competitive advantage. The areas design innovation and innovation strategies are considered out-of-scope for our work. In late 80s, there were some researchers who came up with the role of the designer. The designers generate ideas for possible design solutions. Secondly, they evaluate their alternative solutions with regard to the specified requirements and goals. Finally, they select the most appropriate solution, and communicate their design intent to other people involved with the product development [Cross, 1989].

As a summary of the literature review on design methodology it is understood the current days' core research on design methodology is driven towards expediting the engineering design tasks, especially the analysis and optimization. The exact steps undertaken to complete a design project might vary on case-by-case basis but the general framework of engineering design process has undergone very little changes from Asimov's basic model. In addition, the diffusion of computers, numerical techniques, statistical quality control methods, and manufacturing methods have extended the role of the designer and have necessitated the core engineering design research to focus on developing specialized tools to facilitate engineering design. Section 2.2 presents the details of present days' engineering design process in terms of tasks, design models, and analytical tools.

2.2 Understanding Engineering Design Process

With a detailed review of design literature in design process and methodology we understand that there exists three major stages of engineering design process, viz. conceptual design, embodiment design, and detailed design. These three stages are not discrete, there is certain overlap between stages and concurrent operations are also possible. The degree of overlap between the phases of engineering design is dependent on the product [Parmee, 2001].

2.2.1 Conceptual design

The conceptual design stage is comparable to performing a feasibility study. This process initiates the design by identifying the needs, defining the problem, and exploring a wide range of design alternatives against the preliminary design intent. The conceptual design stage demands diverging, or creative thinking from the designer. The designers' innovation and creativity finds special application in addressing the most uncertain aspects of conceptual design stage. A systematic engineering design calls for identification of the design variables, and their influence on the designed product or system performance. To accomplish this goal, engineers develop predictive models to understand and predict the performance of the system. The model of the system is reduced by a number of assumptions to obtain an approximate mathematical (or symbolic) equation set. Simple computer applications such as spreadsheets can handle such simplified mathematical equation sets and give the designer the preliminary information on the problem. Due to the lack of designer knowledge in this stage, there exists a high risk of the engineering design undergoing major changes. Hence, simplified/approximated mathematical equations that can be solved readily are preferred. The goal is not to identify one best solution instead, the designer is interested in identifying a pool of feasible solutions. More details regarding the conceptual engineering design can be found in [French, 1985].

2.2.2 Embodiment design

This stage focuses on defining the functions of the parts, sub-systems, and the whole system. In this stage, the designers' knowledge of the product has significantly, improved and the risk of undergoing major design changes is substantially reduced. During the embodiment design stage the emphasis is on generating multiple feasible alternative solutions by representing the system using coarse models [Parmee, 2001]. The models here refer to the computer generated CAD models that is used to understand the geometric constraints, assembly dimensions. These CAD models are used to communicate the current design to the product design team that designs subsequent products. If the optimization routines are to be integrated with the analysis tools such as CFD or FEA, the coarse-meshed analysis is undertaken in this stage.

2.2.3 Detailed design

This stage of design identifies a candidate solution that is considered almost stable. A single global solution is chosen from a pool of feasible solutions available from the embodiment design. This brings the designer to the most time consuming stage where the optimization techniques are coupled with the high-fidelity models. Significant computational expense is needed to perform complex engineering analysis, such as, CFD and FEA with fine-sized grids. CFD is a practical tool which can be used to predict the performance of components of the thermal systems or to propose modifications to the original design in rehabilitation or upgrading projects. However, the FEA models are much more commonly used in the design optimization process when compared to the CFD models. Because of the high computational cost many designers skip the integration of CFD models with the optimization due to long time to optimization. Hence, optimization algorithms are generally integrated with lower fidelity models. Engineering optimization that uses high-fidelity analysis models may take several days to several months to get one optimization run completed. As a consequence today, CFD is primarily used to provide insight into a limited number of specific design issues rather than as a design and optimization tool. Hence, one of the goals in this work is to use CFD in the analysis component for a design and optimization tool. In such an optimization tool, designer interaction steers the exploration more quickly to the more fruitful areas of the solution space.

2.3 Computers in Engineering Design

Computers play a significant role in the engineering design process. The productivity of the designers has significantly improved due to the development of computer-aided design tools. Initially, the prime objective for the development of the computer tools was to automate the more routine and tedious tasks involved in the engineering design [Birmingham et al., 1997]. However today, the computers have become indispensable as they are employed in assisting the designer in the entire design process from problem selection to manufacturing, and all the way to the end-product performance. Since this research work focuses on engineering design the transformation the computers have brought in to this area is discussed in detail in this section. As discussed in Chapter 1, the tasks applicable for most engineering design problems include problem definition, initial solution development, modeling, analysis, and optimization. Throughout all the tasks computers are finding a spectrum of applications. The role of computers is more prominent in modeling, analysis and optimization. While in problem definition, and initial solution development the designer plays the important role. Researchers in the area of engineering design focus on developing solutions to maximize the use of computers to expedite the modeling, analysis, and optimization processes.

2.3.1 Computers in model generation

Engineers generate a variety of models to assist them in the design process. Gajda and Biles [1978] have classified models as static or dynamic models, deterministic or probabilistic models, and iconic-analog-symbolic models. A static model is one whose properties do not change over time. Dynamic model considers the time-varying effects within the system. Deterministic models describe the behavior of the system wherein the outcome of the event occurs with certainty. In systems, in which the outcome of events is not known with certainty, probabilistic models are used. Iconic models represent the system as it appears. Sketches, engineering drawings, maps, photographs all fall into this category. Analogic models represent certain specific features of the design. Schematic diagrams and flow diagrams are classified under category, for example shear force and bending moment diagram. Symbolic models describe the design using words, numbers, or mathematical equations. For example, the computer models are based on mathematical equations. Using computer software (or symbolic models) the iconic and analogic models can be created. Before the advent of computers the term “model” often referred to a physical prototype. After the inception of computers the term “model” more often refers to a symbolic model. A model may either be predictive or descriptive. A descriptive model enables the designer to communicate the details of the system and to understand the system better. A predictive model is used primarily in engineering to understand and to predict the performance of the system [Dieter, 2000]. The selected list of advantages of computer generated models include evaluating a design by simulating certain aspects of its performance, assisting the designers in the earliest stage of the design especially when the problem is getting defined and concepts generated. Models are used as a mode of communication between the designer, analyst, manufacturer, and customer.

2.3.2 Computers in analysis

During the analysis stage a designer compares the performance of the candidate solutions against the design intent. When a large or complex system is designed for the first time the designer may not have complete theoretical knowledge on the system. Hence, the two most common approaches of analysis are experimental and numerical analysis. The experimental analyses require the development of a physical prototype that is built per the design specification. The designers' curiosity or "what-if" analysis cannot be readily done as the physical prototype has to be modified or developed afresh every time. Although this process is common in the development of new concept power plant and other complex large scale projects, it is time-consuming and expensive. Today, the numerical or computational tools are rapidly replacing the physical prototyping. The application areas of computer within the engineering design include computer-aided drafting, solid modeling, numerical optimization, simulation, and analysis. The prime application area of computers in engineering design is in computer-aided design (CAD). In an interactive process between the humans and computers an electronic version of the product gets developed. This virtual model of the product is evaluated using computer-aided engineering (CAE) analysis and simulation tools such as FEA, and CFD. Upon successful evaluation the designer may either build a prototype for further testing or sign-off the design to the production [Birmingham et al., 1997]. However more and more design work being completed without physical prototypes. Research work by Simpson et al., [1998] focused on building polynomial approximations for the computationally expensive analysis by applying design of experiments, response surface models, and regression analysis. The research work of Simpson was built upon the research work of Chen et al., [1996] that developed the robust concept exploration method. The robust concept exploration method was developed to expedite the analysis of design alternatives, and identify the important design drivers.

2.4 Virtual Reality in Engineering

The development of modern computers has tremendously extended the scope of computers in engineering. The prime area where the high speed computers equipped with sophisticated graphical devices and interactive devices finds its application is virtual reality (VR). Though originated in late 1960s the full capability of the VR is still being realized. At present, VR is attracting more and more research attention due to its capability of visualizing geometric models, complex mathematical datasets across engineering, sciences, and business. The scope of this section is restricted to the role of VR in engineering. Engineering education, prototype development, design visualization, engineering analysis, and conceptual product development are the areas where VR technologies are widely used. VR is used to construct a user-centered, three dimensional environment in which abstract and complex information is visualized in an intuitive and realistic manner. Because of this VR is becoming an important tool particularly in engineering design.

Engineering design is a non-trivial, iterative task that involves several design changes before a final decision is made. Display of information is very important to quickly understand the practical difficulties or advantages of implementing a particular design. Development of 3 dimensional CAD models was considered the preliminary step in engineering design. In 1980s, these CAD models were considered superior to the 2 dimensional drawings that were used to represent the engineering parts and assembly. When the large and complex engineering systems were designed the visualization capabilities offered by VR technologies were sought. Thus, VR was initially used for real-time visualization of the 3 dimensional CAD models. Using advanced immersive virtual reality environments, the designer was able to "walk through" a virtual power plant, to understand, visualize and communicate the design issues.

Prior to mid-1990s the CAD models prepared externally were imported into the VR system for visualization. With a goal of combining the CAD model development and design visualization VR technology was used to generate CAD models internally. JDCAD used a 6 degree of freedom wand to generate product models using the shape and positions of the primitives [Liang and Green, 1994]. COVIRDS (conceptual virtual design system) was another design system that created a rough CAD model using voice reorganization and hand tracking-based user interface. This system generated viable CAD model in real-time for the conceptual design phase [Dani et al., 1997]. In addition to the CAD model development for the conceptual design, the VR was used to develop and simulate the performance of a variety of spatial mechanisms. Vance and her colleagues developed a series of tools for engineering design, especially in the synthesis of spatial mechanisms. This includes VEMECS [Kraal and Vance 2001], VRSPATIAL [Vance et al., 2002], and VRNETS [Kihonge et al., 2002].

The visualization capabilities of VR were extended to the support activities of engineering design: engineering analysis and simulation. Aukstakalnis and Blatner [1992] developed *Virtual Wind Tunnel* to simulate a flow field which is very difficult to visualize through experimental or numerical simulations. Using the VR the researchers could analyze the effect of the turbulent flow-field by standing inside a virtual wind tunnel. Researchers from Ford Motor Company visualized the air flow pattern to study the air cooling performance on engine components [Deitz 1995; Mahoney 1995]. Full scale car crash worthiness using computer assisted virtual environment was studied by the researchers from General Motors [Ellis, 1996]. Immersive virtual environment was used to study the influence of engineering design changes on the downstream activities especially manufacturing. Manufacturing process simulation was performed using VR to provide insights to product and process development process. The effects of design changes on the manufacturing process were studied. Virtual assembly design environment was developed for assembly planning and evaluation using constrained motion simulation [Jayaram et al., 1999]. Visualizing complex engineering analysis results is an advantage of VR over two dimensional flat screen media. Active research is underway to extend the VR capabilities by integrating it with a variety of high-fidelity engineering analysis models such as CFD, and FEA. CFD data was shown in different visualization methods using IVRESS [Wasfy and Noor, 2001; Wasfy and Wasfy, 2003].

Although innovative, the industry has been slow to adapt the changes, because of the lack of trained manpower and high cost to implement the VR tools. This necessitated the researchers to focus on developing cost effective, portable, easy-to-implement immersive VR tools. In most current virtual reality applications, the data can only flow from the engineering tools to the virtual environment for visualization. With a goal of enhancing the capabilities of the VR a new research area in Virtual Engineering came into the existence. The details regarding the virtual engineering and its impact in engineering design and decision making is presented in Section 2.5.

2.5 Virtual Engineering

Virtual engineering (VE) is user-centered, first-person perspective, three dimensional computer generated engineering environments that seamlessly allow the designer to perform a wide range of engineering tasks. VE environment couples the product models and engineering tools to facilitate engineering design, analysis, optimization, operations, maintenance, training, and disposal. VE can be used to quantitatively and qualitatively identify the innovative design options. In addition to the visualization the VE techniques are used to predict the overall product performance. As shown in Fig. 2.3 the VE product models include geometric models and all the related engineering models, such as CAD. The engineering tools usually are visual analysis, optimization, and decision making tools. Virtual engineering allows designers to walk through the product and observe how it works. The system can also responds to the changes that the designer brings into the system and engages the human capacity for evaluation and decision making. The basic framework for virtual engineering is presented in Fig. 2.3. CAD and geometric modeling forms the foundation of a virtual engineering system. The tools that go along with the CAD and geometric modeling are classified at the first level because they create the visualization characteristic of the virtual product being designed. Multidisciplinary analysis and simulation tools, such as CFD and FEA form the second level which requires interaction capabilities. The top level involves the decision-making/optimization process. These processes cannot be materialized without the product data management and analysis. The interactive visual format of the virtual engineering system help designers develop intuition and understand the product realization better. At all three levels, designer can visualize the product model and data in the virtual environment whenever necessary. In a virtual engineering system, designers can control the geometric and engineering models in a virtual environment, as shown by the double arrowed lines in Fig. 2.3.

VE tools are developed to facilitate the design cost reduction and rapid product realization. Yeh [1997] developed an integrated virtual environment for structural shape design. This includes design specification, sensitivity analysis, and design model manipulation. The design model manipulation was accomplished by introducing a NURBS-based volume around the geometric features. On changing the volume of the NURBS, the product geometry was modified in the virtual environment [Yeh and Vance [1998]. Ryken and Vance [2000] facilitated engineering design process by integrating analysis and simulation tools such as FEA in a VE environment. Designers received feedback on the stress distribution in response to their product geometry modification. VE tools not only help engineers visualize the geometric shapes of the product, but also to understand complex operating conditions, for example, a power plant.

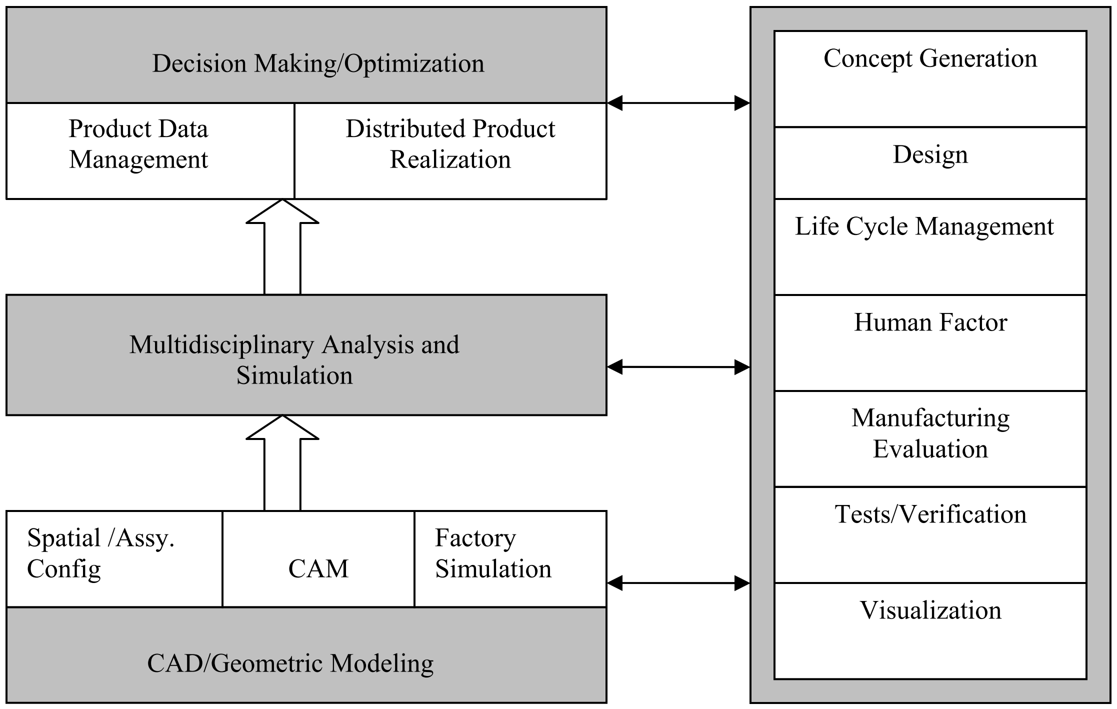


Fig. 2.3 Implementation Framework of a Virtual Engineering System [Xiao and Bryden, 2004]

McCorkle et al., [2003] developed CFD visualization tools that focus on helping engineers resolve product realization problems in an interactive virtual environment. Xiao et al., [2005] has presented the concept of design-analysis integration in a VE framework using VE-Suite. VE-Suite enables the seamless coupling of visualization module, computational engine, and graphical user interface with the high-fidelity analysis models such as CAD, CFD, and FEA in an immersive virtual reality environment. Huang [2006] developed an interactive evolutionary design environment using the VE-Suite framework to facilitate engineering optimization using evolutionary algorithms (EAs) as the optimization algorithm, and Fluent CFD analysis package as the evaluation mechanism for coal piping system design.

The design process uses an iterative approach that allows design changes to be evaluated on-the-fly using CFD analysis.

2.6 Engineering Optimization

Optimization is the process of maximizing the desired quantity or minimizing the undesired one. The term “optimal solution” means the “best” among the feasible solutions for a given set of objective functions, design variables and constraints. Optimization finds its application in engineering during the decision making step [Siddall, 1972]. A variety of optimization methods have been reviewed by Siddall [1979] and those methods were broadly classified under four main areas: optimization by intuition, optimization by trial-and-error modeling, optimization by numerical algorithm, and optimization by evolution. Optimization by intuition is commonly used as a starting place in the design of complex systems. Optimization by trial-and-error arises when the first feasible design is not the optimal one. Before the advent of fast computers this technique was the predominant one. Optimization by numerical algorithm is the area where mathematical theories find its application in engineering. Optimization by evolution is very popular among the optimization methods, especially during the conceptual design phase where the designers’ knowledge about the product or system is limited. Designers prefer wide search space exploration capability using approximate mathematical models. Evolutionary optimization techniques find its application in a wide variety of optimization problems due to its broad search space exploration capabilities.

Optimization theory finds its application in all disciplines of engineering in four major areas: design of component or systems, planning and analysis of operations, engineering analysis and data reduction, and control of dynamic systems [Reklaitis, 1983]. Engineering design is a special case of optimization where a set of objectives have to be optimized while satisfying *functional* and *regional* constraints. In most engineering applications, economic factors such as total capital cost, cost-benefit ratio, etc or technological factors such as minimum production time, maximum thermal efficiency, minimum energy utilization, etc are chosen as the criteria for optimization. The optimization problem modeling a physical system involving only one objective function is called *single-objective optimization*. And that with more than one objective function is known as *multi-objective optimization*. These objective functions are subjected to *functional constraints*, also called as equality constraints, and *regional constraints*, also known as the inequality constraints. Optimization strongly depends on the appropriateness of the mathematical model hence, it is essential to develop the model that includes all the variables and constraints that influence the operation of the engineering system. The generalized mathematical model for an optimization problem can be represented as

$$\begin{array}{llll} \text{Minimize} & \mathbf{F}(\mathbf{x}) & \mathbf{x} \in X & \\ \text{Subject to} & h_j(x) = 0 & j = 1, 2, \ldots, m_1 & \text{(functional constraints)} \\ & g_j(x) \leq 0 & j = 1, 2, \ldots, m_2 & \text{(regional constraints)} \end{array} \qquad (2.1)$$

The terms **F**, h and g are the n-vector objective functions, equality and inequality constraints respectively. Equation 2.1 illustrates an optimization problem with m_1 equality (or functional) constraints and m_2 inequality (or regional) constraints respectively. The term **x** is the vector of design or decision variables $[x_1, x_2, \ldots x_n]$, where X indicates the n-dimensional design space [Papalambros, 1994]. To understand the optimization process and the common terms used in optimization a simple example problem is presented in Section 2.6.1. Optimization in engineering design has become inevitable and it is facilitated with the advent of fast computers. Despite the availability of the powerful computers the optimization still remains to be the most time consuming task in the engineering design. Over the last three decades many researchers have developed algorithms to expedite engineering optimization thereby resulting in rich literature. More details related to engineering optimization can be found in [Siddall 1982; Jaluria, 1998; Deb, 2001; Coello Coello et al., 2002; Hernandez and Fontan 2002]. A survey of most commonly used optimization techniques are presented in Section 2.6.2. The most popular optimization technique, the evolutionary algorithms, is presented in detail in Section 2.7. Many researchers have used interactive techniques to expedite engineering optimization. The role of interaction in engineering optimization is presented in detail in Chapter 3.

2.6.1 Example problem

The terms used in Equation 2.1 are described using this simple example problem on volume optimization of a cylindrical tank. Consider a designer is interested in designing a sheet metal cylindrical tank to store water of volume V. The objective of the designer is to optimize the cost of the tank which directly depends on the amount of the metal used.

Design variables. The first step in the optimization problem is the identification of appropriate set of independent design variables. In this example problem, the design variables are tank diameter D and its height h.

Objective Functions. The objective functions indicate the goal of the optimization problem. Therefore, the objective functions have to reflect all relevant system characteristics. Defining the objective functions is a key issue of importance as the outcome of the optimization is directly related to the quality of the objective functions. In this example problem, the objective function is the cost of the sheet metal which is directly related to the surface area of the sheet metal. The surface area of the tank is given by (A):

$$A = 2\left(\frac{\pi D^2}{4}\right) + \left(\pi D h\right) \tag{2.2}$$

If C is the cost of the metal sheet per unit area, then the objective function (F) can be written as:

$$F = AxC \tag{2.3}$$

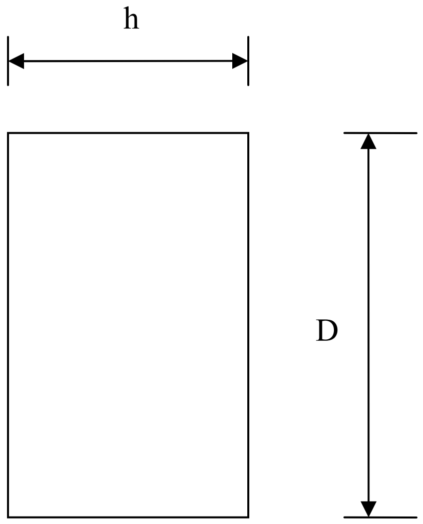


Fig. 2.4 Schematic Representation of a Cylindrical Tank

Constraints. The constraints are classified into explicit and implicit constraints. The explicit and implicit constraints together define the feasible solution space. Explicit constraints are directly specified when optimization problem is formulated. The implicit constraints remain unidentified at least until the detailed design and analysis phase. Implicit constraints exist primarily in the design problems involving continuous variables. The explicit constraints can further be classified into functional (or equality) constraints and regional (or inequality) constraints. The functional constraint for this example problem is that the tank should have a volume V which is given by:

$$V = \frac{\pi D^2 h}{4} \tag{2.4}$$

The regional constraints are imposed by the designer based on the assembly requirements, manufacturing issues, etc. In this problem, the regional constraints are certain preferred maximum and minimum values for D and h.

$$D_{\min} \le D \le D_{\max}$$

$$h_{\min} \le h \le h_{\max} \tag{2.5}$$

To determine the optimum solution for this problem the designer has to choose an appropriate optimization algorithm. The performance of the optimization algorithm is problem dependent. Hence, the designer should have sufficient knowledge on their problem and on the optimization algorithms to select an appropriate optimization algorithm (presented in Section 2.6.2) that suits their problem.

2.6.2 Optimization techniques

General search and optimization techniques are classified into enumerative, deterministic and stochastic methods [Coello Coello, 2002]. Enumerative techniques are exhaustive point-by-point domain search techniques that are inefficient and infeasible for large search spaces. Deterministic algorithms attempt the search process using problem domain knowledge unlike the exhaustive enumerative techniques. Greedy algorithms, hill-climbing algorithms, branch & bound, depth-first, breadth-first, and gradient based methods are all classified under deterministic approach. Gradient-based methods are most common among deterministic techniques. This technique is an analytical approach and is applicable to continuous, twice-differentiable functions. If the optimization involves a large number of design variables, it can be time consuming to obtain the descent direction and the step size needed to carry out the optimization process. Deterministic methods are often less effective when applied to NP-Complete or high-dimensional problems due to the need to have problem domain knowledge to limit search space [Garey and Johnson, 1979; Goldberg, 1989; Fogel, 1999; Michalewicz and Fogel, 2000]. The common difficulties in using gradient based optimization techniques include: dependency on the mathematical models, strong dependence of convergence on the chosen initial solutions, getting stuck to the sub-optimal solutions, inextensible to all problem types, infeasible for discrete search space, and inextensible to parallel computing [Deb, 2001].

Stochastic techniques are preferred to enumerative and deterministic techniques for engineering optimization problems. Many engineering problems are high-dimensional, discontinuous, multimodal and/or NP-Complete. These types of problems are termed irregular [Lamont, 1993]. Because of the difficulty of applying deterministic methods to irregular problems, generally stochastic methods are used. The stochastic techniques are more robust in searching the global optimum than deterministic techniques and are faster than enumerative methods. A selected list of stochastic techniques include simulated annealing [Kirpatrick et al., 1983], evolutionary algorithms (EAs) [Goldberg, 1989; Michalewicz, 1996; Back, 1996] and Tabu search [Glover and Laguna, 1997].

Simulated Annealing. The concept of simulated annealing is based on how metals re-crystallize during annealing process. Initially the annealing process starts with a disordered liquid at a high temperature. This system is slowly cooled to reach the thermodynamic equilibrium. As the cooling proceeds, the system returns to more ordered and frozen ground state. The initial state of the thermodynamic system is analogous to the initial solution of the optimization problem. The energy equation of the thermodynamic system is analogous to the objective function, and the ground state is analogous to the global optimum. Simulated annealing picks up a random solution and moves forward if the selected solution improves the current optima. If not, the algorithm accepts the solution with a probability value. This probability value decreases exponentially with time or with the amount by which the current optimum proceeds towards the global optima. More details regarding the simulation annealing can be found in [Coello Coella, 2002].

Tabu Search. This is an example of stochastic optimization technique that is classified under local search method. This method keeps track of visited solutions and the paths that were used to reach the solutions. This information prevents the algorithm from exploring the search space that contains sub-optimal or fruitless solutions. Generally this technique is used in tandem with other optimization methods [Glover and Laguna, 1997]. Tabu search uses a local or neighborhood search procedure to iteratively move from a solution x to a solution x' in the neighborhood of x, until some stopping criterion has been satisfied. Tabu search modifies the neighborhood structure of each solution as the search progresses. The solutions admitted to $N^*(x)$, the new neighborhood, are determined through the use of special memory structures. The search now progresses by iteratively moving from a solution x to a solution x' in $N^*(x)$. The search space that may normally go unexplored by the other local search procedures can be explored using this technique.

2.7 Evolutionary Algorithms

Evolutionary algorithms (EAs) are the biologically inspired optimization technique that blends a pair of solutions to create better solutions. EAs are based on the Darwinian concept of “survival-of-the-fittest”. EAs are the most commonly used stochastic search optimization technique. Due to the robustness and wide applicability of this technique many researchers have proposed optimization schemes based on evolutionary algorithms. A wide variety of advantages is offered by the evolutionary optimization techniques. EAs do not require derivative information of the objective function in order to find the optimum solution. During the optimization, EAs explore the entire feasible space and search from different design points in one run; thus, the probability of finding a local peak instead of the global peak is reduced significantly. Most real-world engineering optimization problems involve constraints, multiple conflicting objectives, with large number of decision variables, and ill-defined design intents. As the number of decision variables increases the search space widens. Exhaustive search methods are usually slow when it comes to larger search spaces. Most engineering design problems involve multiple conflicting objectives that do not have one best solution. The multi-objective optimization problems result in a number of trade-off optimal solutions. EAs are also an attractive method for multi-objective design applications being offering "pareto optimal sets" instead of a limited single design point traditionally provided by other methods. A selected list of engineering problems where EAs were used include piping layouts [Goldberg, 1983], shape optimization of a pneumatic, low-head, hydropower device [Parmee, 1990], shape optimization of finned dissipater [Fabbri, 1998], heat transfer optimization in a cook-stove [Bryden et al., 2003], optimization of high-speed direct-injection diesel engine [Lee and Reitz, 2003], combustion optimization in the low-temperature diesel combustion regime [Yun and Reitz, 2005], and coal-piping system optimization [Huang, 2006].

EAs consist of a population of encoded solutions manipulated by a set of operators and evaluated by some fitness function. There exists a deep level of similarity between the engineering design process and the EAs. Understanding this similarity can help the engineering designer and the EA designer. Engineering design and the EAs are both iterative process. In the beginning of engineering design process, the designer explores a wide variety of design solutions as defined by the constraints. The knowledge of the designer increases as they progress through the engineering design process. Similarly the initial population of the EAs consists of solutions that may represent the feasible region of the design space. Build upon the best solutions from the pairs of solutions as the evolution proceeds. Engineering designer verifies their current design against certain specifications or design intent, if the design does not meet that specification then the designer moves on to the next iteration. Similarly, in the EAs, the current best solution is compared against the pre-specified stopping criteria, if the answer were to be true, the EA process would stop, if not, the EA process would begin from the beginning, as shown in Fig. 2.5.

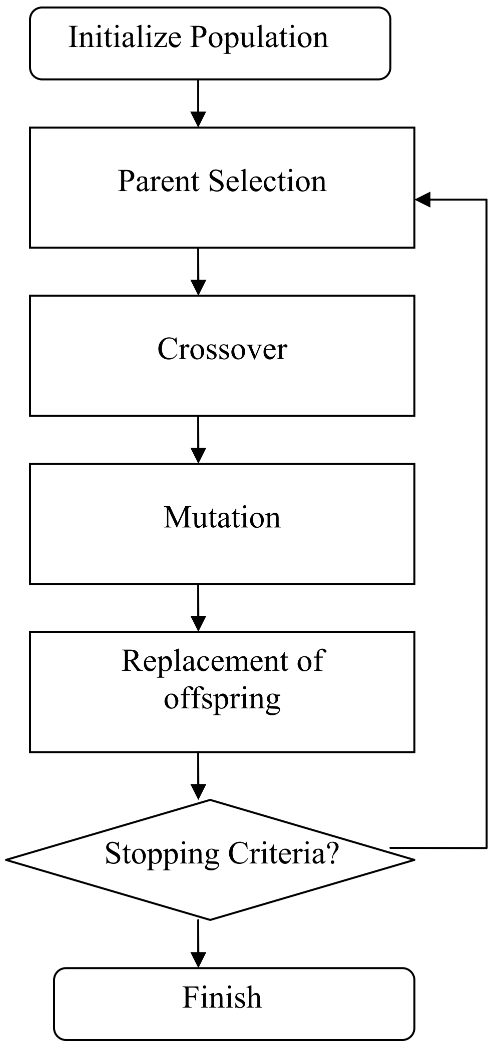


Figure 2.5 Flowchart of Evolutionary Algorithm Process [Cantu-Paz, 2000]

During the conceptual design stage the engineering is not completely aware of the problem-in-hand. Preliminary analysis is conducted to gain some expertise regarding the product. A wide range of engineering solutions are explored during this stage. The EAs start with a population of initial solutions that are not exactly the optimum solutions of designers' interest. EA designer conducts some experimental runs to understand the fitness landscape of the problem and evaluates the efficacy of the solution method. During this preliminary numerical experiments the constraints are soften, and out-of-the-box exploration takes place. Engineering designer gains some insight about the problem from the preliminary analysis and devises a methodology to transfer the concept to reality. The engineering designer now enters the embodiment design phase where the individual components, sub-system, and system dimensions are almost finalized, but the single best solution has not yet been identified. Similarly the EA designer will be aware of the fitness landscape of the problem, the objective function and the constraints. The EA designer then needs to finalize the parameter values e.g., percentage of mutation and crossover, selection and replacement methods. The final stage of the engineering design process is the detailed design where the engineer is interested in nailing down the one best solution which will be implemented for production. Engineers do complete analysis and optimization using appropriate models to ascertain the efficacy of the final solution. Using the knowledge obtained from numerical experiments, the EA designer finalizes the process.

2.7.1 Implementation of Evolutionary Algorithms

For the sake of completeness, this section presents the implementation details of the EAs on the tank optimization problem presented in Section 2.6.1. The implementation details of the EAs presented in this section is also applicable to the test cases demonstrated as a part of this research work. A set of random initial population (say 32 solutions) that covers the entire solution space of the problem is created. The fitness value indicates the performance of the individual solution. The tank optimization is a cost minimization problem. The solution that meets the specified functional and regional constrains using the lowest cost is considered optimum. Every solution (or individual) in the population is represented by a series of design variables as shown in Fig. 2.6. The subscript 1 indicates the solution id within the population. Let D_1 and h_1 represent the diameter and the height of the cylindrical tank. The values of the design variables are chosen such that they meet the regional constraints. Let the volume, surface area, and the fitness value of this solution be V_1, A_1, and F_1 respectively.

D_1	h_1

Fig. 2.6 Representation of a Solution in an EA

The EA population contains a total of 32 solutions each with each solution having a different set of design variables. Just like biological reproduction, a pair of solutions is chosen. The Parent1 is chosen at random, the co-parent (Parent2) is chosen using roulette selection method. The reproduction is achieved using crossover and mutation operators. The children (new solutions) are inserted into the population and the procedure starts over again until the stopping criteria are met. Fig. 2.7 shows the simple crossover operation for this problem. The crossover takes place between the parent 1 and parent 2 as shown the figure to produce child 1 and child 2 respectively. The purpose of the crossover is for the wider search space exploration. Since the length of the solution in this problem is 2, a simple crossover operator is chosen. However, for solutions with longer length random single point or multi-point crossover can be chosen. The mutation operation is performed on the Child1 and Child2 to explore the search space locally. Figure 2.8 shows the single point mutation operation on the Child1.

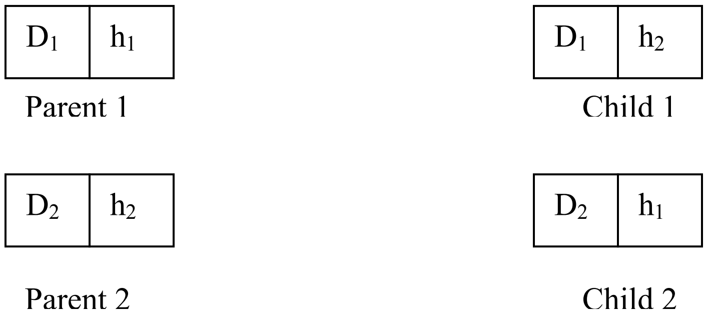


Fig. 2.7 Single Point Crossover

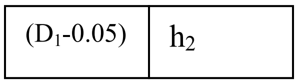


Fig. 2.8 Single Point Mutation

Now the fitness values of Child1 and Child2 are estimated based on their surface areas A_{C1} and A_{C2}, and the volumes V_{C1} and V_{C2}. There exists a variety of ways to perform replacement operation. In this research work, we used *roulette* replacement, i.e., the child with higher fitness values will replace their parents in the population. In summary, the parent selection, crossover, mutation and replacement operations make up one mating event. EAs would stop either if the pre-specified mating events are completed or if the fitness value of one or more solutions in the population equals the optimum value of the problem. For more details on the parent selection, types of crossover, mutation and replacement refer to [Holland, 1975; Golderg, 1989; and Ashlock, 2006].

EAs are preferred optimization techniques due to their advantages. However, the disadvantages of the EAs cannot be undermined. The disadvantages of EAs include: loss of diversity resulting the determination of sub-optimal solutions, and long time for fitness value evaluation when the computationally demanding CFD analysis are coupled with the engineering optimization The issues due to the loss of diversity were addressed by imposing geography in the form of a combinatorial graph on an evolving population. The graph-based evolutionary algorithms (GBEAs) mimic the behavior of many constraints on mating observed in biology [Ashlock et al., 1999]. A combinatorial graph or graph, G, is a set of V(G) of vertices and E(G) of edges where E(G) is a subset of the unordered pairs that can be drawn from V(G). Two vertices of the graph are neighbors if they are members of the same edge [West, 1996]. Within this population structure the solutions are held on the vertices of the combinatorial graphs. GBEAs restrict the rate of information spread within the evolving population by permitting the reproduction only between the creatures that share a common edge. This restriction imposed by GBEA increase diversity and prevents premature convergence to a sub-optimal solution space [Ashlock et al., Accepted]. GBEAs have been used on a real-world engineering challenge, optimization of temperature distribution in a third world cooking stove [Urban et al., 2002]. The most commonly used combinatorial graphs are shown in Fig. 2.9.

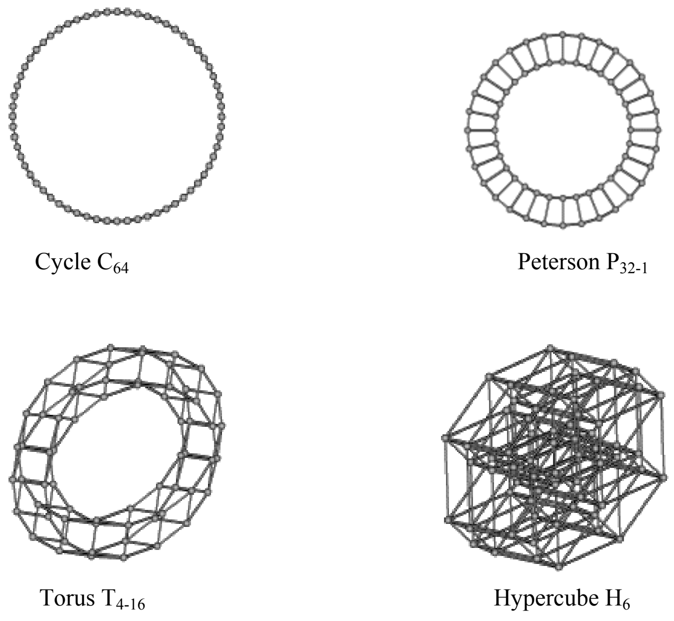


Fig. 2.9 Examples of Cycle, Petersen, Torus and Hypercube Graphs [Karthikeyan 2003]

The longer time for fitness value evaluation when high fidelity analysis models like CFD and FEA models are coupled can be addressed by introducing designer interaction. The designer interaction can narrow down the search space to a manageable size and thereby reducing the number of futile calls to the high fidelity solvers. To develop a “smart” evolutionary optimization technique, i.e., the evolutionary algorithms guided by designer interaction, a thorough understanding of current interaction methods is necessary. Section 2.8 describes the current interaction methods and the research work done by on interactive engineering optimization.

2.8 Interactive Engineering Optimization

Most real-world engineering optimization problems are complex due to the large number of decision variables, and multiple (sometimes conflicting) objective functions. The researchers in the area of multidisciplinary design optimization (MDO) propose breaking down, or decomposing the large, complex problems into smaller components problems that can be optimized independently. The most logical breakdown of problem is by their disciplines [Bloebaum, 1992]. However, the focus of this research is not on decomposing the problems, but on expediting complex engineering optimization by designer interaction. The complex systems need not be dimensionally very large, it may include an object as small as microchip. The designer should have a thorough understanding of internal relationships between the design variables, constraints and objective function to design them better. If the number of objective functions is large, then the optimization time cannot be reduced without designer preference articulation. As discussed in Chapter 1, the *a priori* and *posteriori* preference articulation requires human intervention either before or after all the optimization runs are done. Hence, these two types of preference articulation are classified under unguided optimization with one way coupling to human preference. The designers' knowledge is not used to guide the optimization process. In contrast, the progressive preference articulation incorporates the human intelligence in the optimization continuously leading to guided optimization process, two-way coupling. Designers' articulate their preferences periodically to control the optimization and thereby restricting the search only to the fruitful areas.

2.8.1 Unguided interactive optimization

The unguided optimization is sometimes preferred by the designer to come up with smart initial guesses (or solutions) for a well-defined engineering design problem. This is the reason a variety artificial intelligence based automatic or self-learning algorithms are being used with the designer providing their preference before or after the optimization runs.

However, the term automatic or unguided optimization may not be applicable for a complex system where designers input are considered critical. Most engineering optimization problems use evolutionary optimization techniques due to its wide search space exploration capabilities. Initially, the designers' intent is to come up with a broad range of solutions. After the evolutionary optimization runs are complete the designers' preferences are used to select the optimum solution. The engineering optimization with *posteriori* interaction are very common hence, in this section, a selected list of research work done on unguided interactive optimization using *a priori* preference articulation is discussed.

In the a priori preference articulation the designers' prior knowledge is used to restrict or limit the search to a narrow region. After prescribing their preference initially the designer initiates a computer to come up with an optimum solution. The physical programming (PP) method converts a multi-objective problem into a single objective problem by using preference functions that capture the designer's preference [Messac, 1996]. Some of the limitations of the physical programming method include: requirement of *a priori* selection of parameters for each of the objective functions, provides information for only one design scenario, provides no information about the Pareto designs in the neighborhood of the current design. Deb [1999] uses an analogy from goal programming and allows the designer to define a goal (a single desired combination of characteristics) towards which the search is directed. Yukish and his co-workers proposed the goal programming approach for the multidisciplinary design optimization problems. A novel method of collaborative optimization was implemented using goal programming approach to facilitate multidisciplinary design and optimization at the parts, sub-systems, and the system level [McAllister et al., 2000]. The goal programming approach essentially converts a complex problem to a single objective problem. This problem can be considered a typical characteristic of *a priori* interaction.

Cvetkovic and Parmee [1999] allow the designer to articulate fuzzy preference before the optimization runs. These fuzzy preferences are turned into specific quantitative weightings. Each criterion gets a weight w_i and a minimum level of dominance τ. Based on this information the unguided optimization proceeds to find out an optimum solution. The designer has no intermediate control on these search algorithms. *A priori* preference articulated multi-objective evolutionary algorithms allow the designer to specify the maximal and minimal acceptable weights. These weights are the trade-off for one criterion over the other. Based on the trade-off information, the maximum and minimum utility functions are constructed. This *a priori* information is was used to move the EA towards Pareto-optimal solutions. This technique was applied on four standard test problems, each with two objective functions. The results indicated that the *a priori* preference articulated, algorithm explores the search space much better and converges much faster towards the optimal [Branke, et al., 2000; 2001].

2.8.2 Human guided interactive optimization

The inspiration for most interactive optimization evolved from the computational steering research. Kraemer and Vetter [1998] defined computational steering as, “the online management of the execution of an application and its resources for the purpose of either performance improvement or application exploration.” Computational steering is a broad research area that requires integration of techniques from a wide range of computing disciplines, including human computer interaction, graphics, visualization, parallel and distributed systems, and performance evaluation. The computational steering allows the designer to interactively control the problem parameters during the runtime. The research in the area of computational steering is done with an assumption that the designers' knowledge would help to proceed towards the final answer rapidly [Winer and Bloebaum, 2001]. Based on the concept of computational steering many researchers have developed computational steering environments that include VASE [Jablonowski et al., 1993], SCIRun [Parker and Johnson, 1995], and CUMULVS from Oak Ridge National Laboratory [Geist II et al., 1997]. These computational steering systems require expensive computer hardware and networks to run and store the vast amounts of data generated. This situation motivated researchers [Winer and Bloebaum, 2001] to come up with visual design steering (VDS). VDS was developed for multi-criteria design optimization. VDS is said to be implemented at any point during the design process. VDS uses approximate methods with reasonable level of accuracy required by the designer to visualize the datasets even on personal computers. Both the VDS and the computational steering approach acknowledge the importance of visualization. Visualization provides the designer with the information on the current stage and behavior of the system. This also enables the designer to understand the impact of their parameter changes. As a part of VDS graph morphing is proposed. Graph morphing allows the designer to represent the n-dimensional optimization problem in two or three dimensions. The most critical design variables are placed in these axes and the remaining design variables are placed in graphical "switches".

This research work is interested in developing human-guided interactive optimization technique to expedite evolutionary optimization in complex engineering problems. In this Section, a general overview of computational steering methods was presented which lead to the understanding of importance of visualization in interactive optimization. The rest of this section is dedicated to study a selected list optimization methods based on evolutionary algorithms and that are equipped with continuous designer interaction and/or visualization capabilities to expedite optimization. The interactive physical programming (IPP) framework was developed by [Tappeta et al., 2000] to overcome the limitations of the physical programming methodology. IPP takes into account the designer preference during the optimization process in addition to providing the designer with the Pareto-sensitivity information, Pareto surface representation using response surfaces, trade-off analysis and decision making capability, and a Pareto visualization tool for trade-off studies. The interactive evolutionary multi-objective optimization (I-EMO) proposed by Deb and Chaudhuri, [2005] involves a decision-maker in the evolutionary optimization process and helps choose a single solution at the end. I-EMO first determines a non-dominated Pareto-optimal front using an evolutionary optimization technique. The designer supplies the limiting trade-off values to arrive at the partial Pareto-front. From the partial Pareto-front the knee solutions are computed. The optimal solutions may sometimes be sensitive to local perturbations. Even a small change in the decision variables will influence the overall performance. The optimal solutions should be stable against small disturbances because in practice a solution is difficult to implement exactly with an infinite precision. Thus, the designer is often interested in robust solutions that are relatively insensitive to variable perturbation. Huang [2006] developed a multi-threaded interactive evolutionary design environment to achieve optimum coal pipe design. The optimization run proceeds in a model thread, while the designer can make certain changes to the existing solutions in a view thread and visualize the results. The multi-threaded interactive evolutionary design environment was used to expedite the engineering optimization.

2.9 Summary

This chapter presented the current trends in engineering design in the light of high fidelity models and advanced computing technologies such as virtual engineering environments. In addition, the Sections 2.1 - 2.5 pointed the research direction towards rapid product development by expediting the engineering design process, in particular, the engineering optimization process. Sections 2.6 and 2.7 presented the role of optimization techniques in general and the unmatched supremacy of evolutionary optimization techniques on engineering optimization. Thus the stage is set with the availability of a variety of optimization tools, high fidelity models, and advanced engineering environments. One of the major factors that slow down the evolutionary optimization is the longer time to compute fitness values when the high fidelity analysis models are integrated with the optimization. From the background research done so far, it is understood that interactive optimization system is the solution to design problems involving complex systems. Section 2.8 was dedicated to provide the details regarding unguided (or computer controlled) interactive optimization and human guided interactive techniques. The material outlined in this Section opens up Chapter 3 where further discussion on this subject matter is done and the need for the progressive designer interaction is explained.

CHAPTER 3. CURRENT TRENDS OF HUMAN INTERACTION IN ENGINEERING DESIGN

This chapter focuses on the present days' role of human interaction on engineering design. The current interaction methods discussed briefly in Section 2.8.2 are revisited in this chapter to present their similarities and the differences with respect to this research.

3.1 Human Interaction Systems in Engineering

Human-guided interaction schemes are often well-suited for engineering problems involving complex systems. This understanding has resulted in the development of several human interactive methods in engineering especially in the most critical areas of analysis, simulation, and optimization. Anderson et al., [2000] pointed out the importance of interaction on engineering optimization problems. The successful implementation of an optimization algorithm is proportional to the designers' understanding of the problem and the level of design interaction with the problem parameters. More and more researchers agree upon the importance of graphical visualization to facilitate the designers' understanding of the solution, state of the problem, and behavior of the design variables [Kraemer and Vetter, 1998]. The entire research area of computational steering was emerged based on the role of visualization to make design changes to the parameters rather than letting the computational algorithm complete the runs. The basic assumption of computational steering is that the experienced designer/analyst can steer the engineering design process to solution more rapidly. In contrast, if the solution strays to fruitless areas, the designer can either redirect the solutions or start off with the new initial conditions. Computational steering research is based on the paradigm that the simulation and visualization cannot be decoupled. Hence, the concept of computational steering is leaning more towards the online visualization, i.e., the designer is enabled to interact with the solution by seeing the parameters during the analysis or optimization runs. Also the computational steering research supports the use of full datasets without any approximations to speed up the process or reduce complexity. A selected list of research work based on computational steering include [Parker et al., 1995 and 1997; Beazley and Lomdahl 1996; Longacre et al., 1996].

SCIRun, a scientific programming environment for computational steering was developed by [Parker et al., 1995]. SCIRun was originally intended to solve specific problems in computational medicine, but its scope was extended to serve as a generic problem solving environment in computational sciences and engineering with the emphasis on steering large-scale scientific computations. This system allows the designer to enter complex problems in the form of large equation sets. These equation sets are then sent to the computational solver. At any time, the designer can change the design variable values and implement them in the running analysis. Despite all the advantages of this system this system may not suit an engineering optimization problem connected to the conceptual design stage, where the designer may not be completely aware of the system. Hence, the mathematical equations sets cannot be prepared upfront. Multi-function optimization and visualization environment (MOVE) was developed by Longacre et al., [1996]. The solutions of the optimization problem are represented in a visual two dimensional form. The design variables are changed by the user using the graphical user interface. This tool also presents the designer the sensitivity information as line graphs and Pareto information in a simple manner to enable the designer understanding of the optimization problem. Despite the advantages the MOVE tools become really complex when a large number of design variables are to be studied. There exists a variety of tools are built based on computational steering paradigm, their differences are however very small and are mostly based on how they manage the large datasets that are neither reduced nor approximated.

With the emphasis on the role of visualization in engineering the concept of visual design steering (VDS) was proposed by [Winer and Bloebaum 2002] to cater to engineering design problem, especially those involve multidisciplinary optimization. VDS controls the computationally intensive design processes using visualization. To steer the design process to solutions more efficiently the visualization can be implemented before, during, or after the design process. This is the area where the VDS differ from the computational steering philosophy computational steering philosophy emphasizes on visualization during the design (or optimization) process. In addition, the computational steering methods require large data transfers, sophisticated networking capabilities, and high-performance computers. The VDS paradigm accepts reduced or approximated datasets to maintain the interaction in the real-time or near-real-time. VDS can operate on a range of computer hardware from powerful workstations to a personal computer. VDS addresses the issues related to multi-dimensional datasets by graph morphing, i.e., representing the most critical design variables in two or three axes and the remaining variables can be visualized by enabling graphical switches. The major advantage of VDS scheme is that the design variables may be changed in the real-time. In their work on method validation for the graph morphing the researchers have demonstrated the usefulness of the variable, and analyzed the constraint limits and redundancy. More details regarding the implementation of VDS, graph morphing and their role in facilitating multidisciplinary optimization problems can be referred from [Winer and Bloebaum, 2002 (a) and (b)]. Despite several advantages of this system over the computational steering methods, this scheme doesn't give much emphasize on the most common evolutionary optimization techniques and the use of high fidelity analysis models.

Many researchers have worked to perform engineering tasks from the virtual environment by coupling engineering analysis and graphical visualization. SphereVR the first VR environment for the design of spherical four bar mechanisms was developed by Osborn and Vance [1995]. This system places the designer in the virtual environment to virtually built and test prototypes to develop an optimum mechanism. The graphical representation of the parts in the virtual world gave the designer the freedom to move along all three axes to study the impact of the design changes. Boilermaker developed by Diachin et al., [1996] is another earliest application that coupled engineering analysis with graphical visualization. The computational model of an industrial furnace placed in a virtual environment enabled the designer to study different furnace configurations and choose the design variables to create an optimum design. Yeh and Vance [1997] used the sensitivity-based structural design system that allows the designer to interact with the structural elements and systems in an immersive virtual environment to perform optimization to obtain the stress levels in the structures within the desired range. Ryken and Vance [2000] facilitated engineering design process by integrating high fidelity analysis and simulation FEA tool in a VE environment. Designers received feedback on the stress distribution in response to their product geometry modification. Another example of coupling engineering analysis and virtual environments is DN-Edit [Kihonge et al., 2002]. DN-Edit allows NURBS-based (Non-Uniform Rational B-Splines) surface geometry to be altered interactively. A variety of interaction were possible using virtual cursors that allowed interaction with geometry surfaces, enabled surface points to be displaced, material to be added or removed. The resulting NURBS-based surface can be exported to various CAD or analysis programs. The shaping of three-dimensional geometry in a virtual environment bypasses the obstacles and limitations of a two-dimensional human computer interface. All the methods discussed in this paragraph use datasets that are reduced or approximated.

3.2 Scope of this Work

From Section 3.1, it is evident that including designer in the loop expedites optimization, and enhances designer understanding of the problem. Especially during the conceptual design phase the problem formulation takes place as a part of optimization process. Therefore, it is desirable and beneficial to optimize an inexact or ill-defined problem using designer interaction. From the background research done so far, it is understood that visualization enabled interactive optimization system is the solution to design problems involving complex systems. The interactive exploration of the search space with "what-if" analysis may help designer to more effectively generate promising solutions and to assess the feasibility of the generated solutions for implementation. A variety of interactive optimization systems equipped with the state-of-the-art graphical visualization are present both as commercial and open source software. Some of the systems are implemented without the capability of integrating the high-fidelity analysis models during run-time. Especially when thermo-fluids systems are designed the role of optimization integrated with high fidelity models are considered critical. While some immersive virtual reality based designer interactive system use high fidelity models using reductions and approximations, however they facilitate engineering optimization by trial-and-error, and not by evolution. Thus the versatility of engineering optimization is to be considered before designing an interactive optimization scheme. In addition, in some current systems the designer interaction interrupts the optimization or the optimization runs are done elsewhere and the pre-computed results (offline) are visualized in the immersive virtual environment. Despite the availability of power visualization in such systems, the designer choice of parameter values can't be changed in the runtime resulting in the designers' not understanding the implicit design constraints. These obvious gaps in the current human-guided interactive systems can be addressed by the proper understanding of modes of human-guided interaction, current trends in interactive optimization using high-fidelity models, and advanced engineering framework, virtual engineering.

The goal of this research is to study the impact of designer interaction on expediting engineering optimization. Vladimir et al., [2002] summarized the deficiencies of the current optimization software, they are, lack of user familiarity requiring immense training on the optimization concepts, cost of optimization software, and large computational resources required to perform general purpose optimization. In addition to addressing the observation of Vladimir, this research is also driven to simplify the tasks of the designer while interacting with the optimization system. Three simple modes of designer interaction are inspection, modification, and user controlled re-optimization. Depending on the complexity of the optimization, the time span of each interaction mode varies. For example, in a small optimization problems, the designer can shift between the three modes of interaction quickly, whereas, the same cannot be possible in large problems. Large scale interactive optimization poses challenges regarding modifiable visualizations, user friendliness, and algorithmic performance [Chimani et al., 2004]. The goal of this research work is to facilitate large scale interaction on complex engineering optimization problems using evolutionary optimization techniques coupled with high-fidelity analysis models.

Existing literature presents the potential of virtual engineering as a power tool for enhanced productivity and rapid product realization by enabling the designer access a variety of problem solving and decision making tools. This inspires a question, “Why virtual engineering systems are not routinely used?” It is due to the implementation complexity of virtual engineering system. The implementation of virtual engineering requires knowledge of the application and visualization including knowledge on VR, user interfaces, data communication, and their own application. In addition, end-users, e.g., design engineers are not programming experts. Virtual engineering applications can be constructed from scratch, but as with any construction task, applications can be constructed more easily and efficiently by integrating an existing application; the existing application could be automated, or at least a higher lever user interface to assist in the annotation of the application is provided. The general virtual engineering environments that contain software tools should be able to provide higher level functionality on aspects common to arbitrary virtual engineering applications. The existing research on virtual engineering discussed above shows that the construction of the virtual engineering application is an underdeveloped aspect; existing systems can only support specific applications. In addition, existing systems are not tailored to the specific requirements of an effective virtual engineering application. These deficiencies led to the development of VE-Suite for a general purpose virtual engineering environment that fills the gap in existing systems. Thus for this research the interaction framework is provided by VE-Suite, evolutionary algorithms are used as the standard optimization technique, CFD models are the high fidelity models that are coupled with the optimization tools. Since the proposed research work has the characteristics of visual design steering proposed by [Winer and Bloebaum, 2001] and computational steering paradigm with simplified designer interaction. Hence, the proposed interaction is termed as human-guided progressive interaction technique. The feature that makes the proposed interaction scheme common to both the paradigms is: continuous and intermittent designer interaction with the optimization system even during the optimization runs. A key aspect of progressive interaction is the ability to see the proposed design in an easily understandable and intuitive way. Simply visualizing the parameters does not allow the designers to choose the best design and guide the process forward. Rather designer need to see the see the current design and understand its strengths and weaknesses. In this way the designer can ensure that complexities are understood and that the full range of engineering solutions can be explored. This can be achieved by using virtual engineering (VE).

This chapter presented the similarities and differences of the current work with respective to the proposed research in general. More detailed documentation of the contributions from this work is present in future chapters along with the test cases used to demonstrate the progressive interaction scheme.

CHAPTER 4. PROBLEM DESCRIPTION

In order to demonstrate the performance of the proof-of-the-concept progressive designer interaction on engineering optimization three test cases were chosen. The first test case is on image segmentation and optimization. A variety of datasets are used in engineering decision making process. This test case considers the role of digital images as decision making objects from geographically diverse locations. The better the image quality the faster the engineering decision making from diverse locations. The quality of the images and the data transfer issues are approached in terms of image segmentation optimization. The designer interaction is supposed to expedite the image segmentation until the image segments with appreciable quality is evolved. The second test case is the shape optimization of finned heat exchanger. This project is of critical importance especially for electronic components where high amount of heat needs to be removed from very small components. This case study is picked to demonstrate the fact that complex systems need not be large systems. The role of designer interaction to evolve the optimum fin profile is the prime focus of this case study. The third test case focuses on demonstration of interactive optimization using commercial high fidelity analysis models coupled with evolutionary optimization techniques. Also, very few designers are keen to perform CFD analysis due to its complexity and larger run times. The goal is to demonstrate the importance of the CFD analysis and the role of interaction to optimize this complex thermo-fluid system. The details regarding all these three test cases presented in sections 4.1 to 4.3.

4.1 Interactive Image Segment Optimization – Problem Description

Decision making using image data is widely prevalent in medicine, geographical information systems, and aerial surveying. Due to ease of representation, analysis, and knowledge acquisition the digital images are more popular than other forms of datasets. The advantages of digital images are obtained at the expense of their larger file sizes. For example, the uncompressed file size of a 24 bits-per-pixel (bpp), true color, 1200 x 1000 pixel aerial survey image is approximately 3.6 megabytes. Maintenance of high volume of such images in their original quality is not practically feasible. Data management issues arise if the images are to be transferred to remote and low-bandwidth areas. For the last four decades, the data storage and transfer issues have called for the development of many specialized data and image compression algorithms. The performance of an image compression algorithm depends on the type of the image. Thus it is important to understand the different types of image.

An image can be classified as bi-level, gray scale, or true color depending on the amount of chromatic information present in each pixel. Continuous-tone and discrete-tone are the classifications of the image based on the color variation between adjacent pixels. In a continuous-tone image, the color variations between adjacent pixels are so small that they are hard to distinguish, e.g., aerial survey image and medical images. Fig. 4.1 shows a 24 bits-per-pixel, true color aerial survey image. Discrete-tone images have sharp color variations between adjacent pixels, for example, a checkerboard. The compression methods used for continuous-tone images often do not handle the sharp edges of discrete-tone images well. This is due to the difference in the feature redundancy in the images. Based on the allowable loss of information, chromatic information per pixel, and pixel color variation, suitable compression algorithms need to be selected.

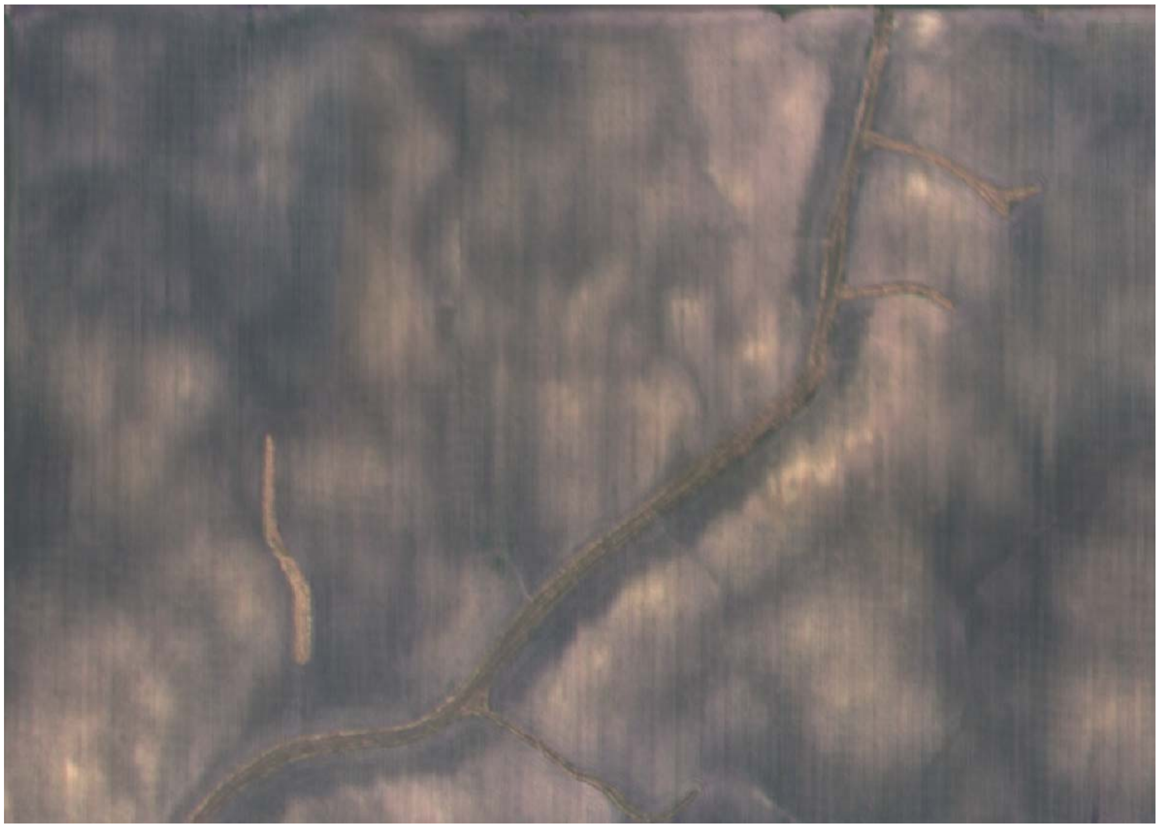


Fig. 4.1 Continuous-tone Aerial Survey Image

Over the last four decades, many researchers have proposed specific image compression algorithms to suit a variety of image types. Thus the image compression algorithms fall into two broad categories: lossless and lossy. A detailed description of a variety of well-documented compression techniques are presented in [Salomon 2004]. Lossy algorithms preserve essential image features and remove image data that is not detectible by human eyes at a desired picture size [Salomon 2002]. These techniques produce high compression ratios that are desired for storing and transferring large images. Lossy compression techniques are preferred if the quality degradation due to compression does not impact decision making. Segmentation-based coding, transformation coding (TC), vector quantization (VQ) and sub-band (SB) coding schemes are commonly used lossy compression techniques [Shukla et al., 2002]. Most natural images can be segmented into regions of high and low details. High detail regions are intrinsically less compressible than regions of low detail. Variable-rate image coding scheme varies the number of bits-per-unit-area according to the local detail. For variable rate image coding applications, segmentation based coding methods generally provide high compression ratios when compared with other compression methods.

Image segmentation subdivides an image into its constituent regions or objects. The level of the subdivision is dependent on the level of details sought by the designer and the nature of the problem being solved. Image segmentation is an essential preprocessing step that determines the eventual success or failure of image analysis, image understanding, pattern recognition, and robotics vision [Mena and Malpica 2003]. Due to the wide applicability of image segmentation, active research is under way to develop faster and sophisticated segmentation algorithms. An early survey on color image segmentation was presented by [Skarbeck and Koschan 1994]. A detailed review of most commonly used segmentation algorithms are presented in [Karthikeyan et al., Submitted]. The basic concepts and the implementation details of most segmentation algorithms are presented in [Gonzalez and Woods 2002; Gonzalez *et al.* 2004]. The performance of the segmentation algorithm depends primarily on the image feature distribution, and pixel color variation. A suitable segmentation algorithm has to be selected from a variety of pixel, edge, region-based, or advanced segmentation algorithms. Region-based image segmentation algorithms are more-appropriate to segment the image feature-by-feature. This type of segmentation expedites the image analysis and decision making.

Despite the availability of a variety of image segmentation algorithms, a number of image segmentation issues have not been adequately addressed. The open issue is that most algorithms use non-optimal segments and the training datasets must be pre-specified. This scenario has motivated our research with an initial assumption that representing an image using optimal number of segments automatically addresses the file size, image quality, and data transferability issues. But, generating optimum image segments is a non-trivial task with the existence of multiple feasible solutions. An automatic, easy-to-implement low-impact image segmentation algorithm is developed as a part of this research work. This algorithm combines the segmenting strengths of weighted Voronoi tessellations with the optimizing capabilities of the evolutionary algorithms. This segmentation algorithm has been used to segment low-variance aerial survey images and a variety of regular photographic images.

4.1.1 Low impact image segmentation

In a low variance continuous-tone image the smallest image distortion can potentially impact the quality. In order to avoid distortions, more image segments are required to preserve the image quality. The larger the number of image segments the higher the computation time. In this study we present a new methodology for developing segmentations based on balanced weighted Voronoi tessellations. The segmentation scheme has the potential to provide significant feature preservation capability that can be equally applied to discrete and continuous tone images. In addition by identifying and grouping various features this process can facilitate engineering analyses. The low-impact image segmentation is implemented by combining the strengths of weighted Voronoi tessellations with the evolutionary algorithms.

Balanced-weighted Voronoi tessellations. Voronoi tessellations (or tiling) [Okabe *et al.* 2000] is a division of a finite subset of the plane into convex polygonal tiles. The tiles are generated by placing a set of tile centers (or generators) into the plane. A *tile* associated with a given tile center *p* consists of all points in the plane closer to *p* than to any other tile center. The boundary between two tiles that meet is always a segment of the perpendicular bisector of the line segment joining the two tile centers. In ordinary Voronoi diagrams, the distances to the point sets are not weighted. Hence, polygonal segments result as shown in Fig. 4.1 (a). Images often contain objects with curved boundaries. For these applications, *weighted* Voronoi tessellations are used to segment the image. A *weighted* Voronoi tessellation differs from a standard Voronoi tessellation in that each tile center has a *weight r* associated with it. The square root of color variance in the locality of the generator is considered the numerical weight *r*. This weight is multiplied by the distance from a generator to a point in the plane when deciding tile membership. A large weight makes distance to a generator more expensive, shrinking the size of the tile. A small weight makes the distance to a generator cheap, increasing the size of the associated tile. The boundaries between weighted Voronoi tiles are no longer line segments. Examine the comparison of squared distances made for two tile centers (x_1, y_1) with weight r_1 and (x_2, y_2) with weight r_2 for a point (u, v) whose tile membership is to be decided:

$$\{ r_1^2 [(x_1 - u)^2 + (y_1 - v)^2] \} < \{ r_2^2 [(x_2 - u)^2 + (y_2 - v)^2] \} \tag{4.1}$$

If $r_1 = r_2$, then all quadratic terms in the comparison cancel resulting in a straight line segment. In contrast if $r_1 \neq r_2$, then quadratic curve boundaries result as shown in Fig. 4.2 (b) [Ashlock *et al.* 2006].

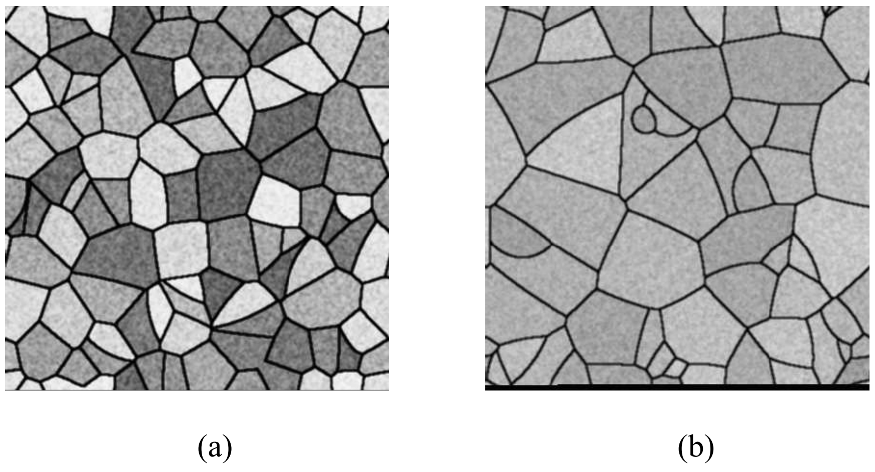


Fig. 4.2 (a) Standard Voronoi Tiling and (b) Weighted Vornoi Tessellations

Evolutionary optimization scheme. An evolutionary optimization technique is used to optimize the image segments generated by weighted Voronoi tessellations. The image pixel data is discrete, multiple good solutions exist this necessitates the use of an evolutionary algorithm (EA). Weighted Voronoi tessellations are easily represented in an evolutionary algorithm as well. Each solution contains the voronoi generators (or point set) and their corresponding numerical weights. Fig. 4.3 shows a single solution in the EA population that is represented using four image segments. The length of the solution is equal to the number of image segments used to represent the image. Hence, we have the pixel values (X, Y) of the Voronoi generator and their numerical weights.

X_1	Y_1	Wt_1	X_2	Y_2	Wt_2	X_3	Y_3	Wt_3	X_4	Y_4	Wt_4

Fig. 4.3 Solution Representation for Evolutionary Optimization

In this study, both the location of the points and the weights are optimized to minimize the color variance within tiles. The choice of EA parameters such as parent selection, crossover, mutation, and replacement was made from experience and small numerical experiments. The first parent was chosen at random and the co-parent was selected using roulette selection. Random two-point crossover was used with probabilistic mutation with a 50% chance of increasing the number of image segments and another 50% chance of randomly selecting 'n' locations and increasing their numerical weights by 10. Designer interaction helps identify the number of distinct image features in a complex, continuous-tone, low-variance images, for example, aerial survey images. Hence, periodic designer interaction with the segment optimization routine expedites segmentation, image understanding, and decision making. The details regarding the implementation of progressive interaction scheme for image segment optimization is presented in Chapter 6.

4.2 Shape Optimization of a Finned Dissipater – Problem Description

Finned dissipaters are commonly used in many engineering sectors, where high heat fluxes must be transferred. This is of critical importance especially for electronic components where high amount of heat needs to be removed from very small components. The optimization of the fin shape is extremely important for the optimal performance of the system that generates heat. Shape optimization of a fin is encountered during the detailed engineering design stage where the physical configuration of the fin plays a very significant role on the flow parameters, such as temperature and flow velocity. The shape optimization is broadly classified into direct and inverse optimization methods. Direct optimization problem is concerned about minimizing an objective function subject to constraints on geometry or flow conditions, as shown in Equation 2.1. The direct method becomes tedious and time-consuming when a large number design profiles need to be analyzed. The inverse optimization method considers an end results, say for instance, pressure or velocity distributions and determines the shape or geometry that gave rise to such pressure or velocity distribution. Both the methods have unique advantages and disadvantages. The two distinct demerits of the inverse problems are: 1. the distribution imposed on the formulation may not be physically realizable, so a solution may be impossible and 2. even if the distribution imposed is physically realizable, it may not be an optimal distribution [Fan and Zhu, 1998].

4.2.1 Evolutionary algorithms for fin shape optimization

The direct optimization problems are becoming more promising option due to the advancement in computer hardware. In this research, direct shape optimization using evolutionary optimization technique is considered. Evolutionary optimization techniques are preferred when the thorough exploration of search space is desired. Fabbri [1997 and 1998] has used an evolutionary algorithm to simultaneously optimize the geometry, base thickness and spacing of heat exchanger fins while maintaining an upper limit on the volume of the fin. The governing equations were solved using the finite element method. Chin-Hsien et al., [2001] solved the inverse shape design problem using the conjugate gradient method. A methodology was developed to design the shape of solid medium based on its energy loss to the surrounding stationary fluid. Suram et al., [2006] explored the applicability of graph based evolutionary algorithms on the shape optimization of the finned dissipater.

In each case, the exploratory capabilities of the evolutionary techniques generate a set of feasible design solutions. However, this advantage was obtained at the expense of longer optimization time. The time to locate an acceptable solution can range from a few days to a few weeks for complex shape optimization. A significant portion of this time is spent on fitness evaluation. The computational solver estimates the initial fitness values of all the solutions in the evolving population and the fitness values at the end of every mating event. A significant time reduction can be achieved if the calls for the fitness value evaluation are done only when there is a progress towards the optimum solution. This scenario necessitates the designer in-the-loop to direct the search into more fruitful regions of the solution space based on his/her knowledge thereby reducing calls for fitness evaluation. Hence with an objective to expedite the complex shape optimization, this problem has been chosen as the test case for the research.

4.2.2 Description of fin setup

Figure 4.4 shows a set of fins. Fluid (water) is pumped through the channel between the curved surfaces of two consecutive fins i.e. along the positive *z*-axis. The vertical surface of the fin is insulated. The fluid velocity and the temperature profiles are assumed to be fully developed. The flow is also assumed to be laminar and incompressible and effects (if any) due to natural convection are neglected. The whole system is assumed to be in steady-state. The thermal properties of the solid and fluid are assumed constant. Fin profiles are to be designed with a consideration that high concavity or convexity may result in manufacturing issues. Taking advantage of the symmetry, only one half of the fin is modeled which is shown in Fig. 4.5. In this Fig. the distance from the base of the fin to the insulated flat plate is assumed to be of unit length. The length of the fin is denoted by *a,* the base thickness by *t* and the spacing between two consecutive fins by *2b*. In Fig. 4.5 due to the symmetry we have the spacing as *b*.

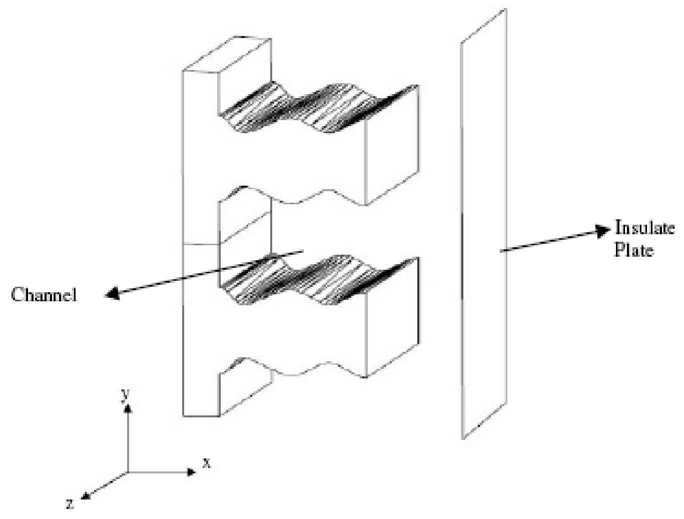


Fig. 4.4 Schematic Diagram of the Fins

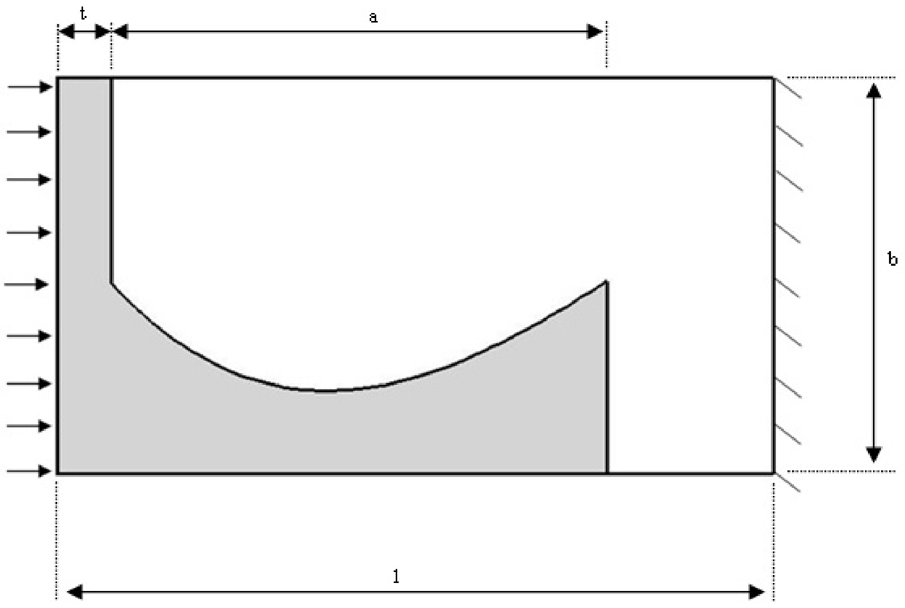


Fig. 4.5 Modeled Fin

4.2.3 Governing equations

The flow of the fluid (water) is given by

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=\frac{1}{\mu}\frac{\partial p}{\partial z} \qquad (4.1)$$

The pressure drop along the *z*-axis is assumed to be constant. It should be noted that the pressure drop is negative in the direction of flow. For boundary conditions, the velocity of the fluid is zero at all of its interfaces with the fin and the insulated plate. The boundary condition $\frac{\partial \mathrm{u}}{\partial \mathrm{y}}=0$ applies on the symmetry boundaries. The temperature distribution in the fin and the fluid are given by Equations (4.2) and (4.3) respectively.

$$\frac{\partial^2 T_s}{\partial x^2}+\frac{\partial^2 T_s}{\partial y^2}=0 \qquad (4.2)$$

$$\frac{\partial^2 T_f}{\partial x^2} + \frac{\partial^2 T_f}{\partial y^2} = \frac{\rho_f c_{pf}}{k_f} u \frac{\partial T_f}{\partial z} \tag{4.3}$$

The temperature of the fluid and the solid are assumed to be thermally fully developed. Due to this assumption any plane parallel to the *x*-*y* plane, the same temperature profiles would be obtained. The fluid temperature variation along the *z* axis is assumed to be constant. The boundary conditions at the non-symmetry plane for the energy equation of the solid and the fluid are given in Equations (4.4 and 4.5) respectively.

$$-k_s \left. \frac{\partial T_s}{\partial x} \right|_{x=0} = q \tag{4.4}$$

$$k_f \left. \frac{\partial T_f}{\partial x} \right|_{x=1} = 0 \tag{4.5}$$

At the interface of the solid and the fluid the heat flux and temperatures have to be equated to conserve energy as presented in Equation (4.6).

$$\begin{gathered} k_f \frac{\partial T_f}{\partial \vec{n}} = k_s \frac{\partial T_s}{\partial \vec{n}} \\ T_f = T_s \end{gathered} \tag{4.6}$$

In Equation (4.6), the term $\vec{n}$ is the direction normal to the surface being considered.

Representation of fin profiles. The fin profiles are represented by a polynomial equation as given in Equation (4.7).

$$y = \sum_{i=0}^{i=n+1} c_i x^i \tag{4.7}$$

In Equation (4.7), the term ‘n’ indicates the degree of the fin profile. The x values are defined by splitting the length of the fin into ‘n+1’ equal points. The terms ‘c_i’ indicate the polynomial coefficients. Based on the degree of the fin profile the designer supplies y_i values. Using the y_i and x_i values the polynomial coefficients are determined.

Fitness evaluation. The fitness of the fin profiles are computed in terms of a non-dimensional number. In order to compute the fitness the temperature at each grid point in the fluid and the solid should be available. These temperature values are divided by the max temperature (T_{max}). Irrespective of the fin profile degree and the cooling fluid parameters the T_{max} corresponds to the lower left corner in the Fig. 4.5. This is because of the same boundary conditions on the left and the lower surfaces of the fin for all the CFD cases. This non-dimensionalizes the temperature distribution. Hence, Nusselt number can be computed using Equation (4.8) as a dimensionless number that is equivalent to the temperature gradient at a surface. The higher Nusselt number is an indication of the enhancement of heat transfer from a surface due to a surrounding fluid, when compared to the case of pure conduction. Thus, the higher the Nusselt number more is the energy transfer and so better is the shape of the fin.

$$Nu = \frac{\partial T_s}{\partial \vec{n}} \tag{4.8}$$

Thus, the optimization problem is one of maximizing the Nusselt number at the lateral surface of the fin.

Interactive shape optimization of finned dissipater. Human-guided shape optimization task is performed to obtain an optimum fin profile. The optimum fin profile is essential to achieve maximum heat transfer through the finned dissipater. The designer’s intelligence was used to develop interactive initial solutions. These “smart” initial solutions potentially reduce the number of fruitless fitness evaluation calls. In addition, these smart solutions “lead” the other solutions in the population in their search for optimum solution. In this proof-of-concept study, we are interested in understanding how the designer created interactive solutions lead the evolutionary algorithm towards the optima when compared to random population.

4.3 Interactive Analysis and Optimization of Mixing Nozzle – Problem Description

Hydraulic mixing nozzles are used to mix the chemical and carrier solutions and store the mixture in a tank. Hydraulic mixing nozzles require pump to provide the fluid flow necessary for the nozzle operation. The prime advantages of the hydraulic mixing nozzles are: high reliability and low maintenance due to the absence of moving parts, low installation cost due to the elimination of propeller and its accessories like shaft, bearings, and drive unit, and easy to clean the storage tank as the propeller and other hindering parts are not used. However, the designer should be aware of certain critical issues before selecting an hydraulic mixing nozzle. Firstly, the mixing efficiency of these nozzles is less when compared to that of mechanical agitators. Secondly, the optimum design and placement of these nozzles are extremely critical to achieve uniform tank mixture in a required time. The designer has two choices to achieve uniform tank mixing, either to use a set of smaller number of nozzles or to develop the optimum nozzle design. The focus of this research is to provide the engineering solution by developing optimum nozzle design.

Fig. 4.6 shows the CAD model of a common hydraulic mixing nozzle that uses a high-speed fluid. The fluid pumped out is send to the nozzle jet, the right side portion of Fig.

4.6. After passing through this region the fluid enters into the open portion called the entrainment portion. The jet is connected to the horn through four supports, as shown in Fig. 4.6. The supports should be thin to avoid overly interrupting the liquid flow around the jet, but must be strong to sustain reasonable loads applied on the horn during cleaning and maintenance. The fluid leaving the entrainment portion passes through the horn that directs the fluid. In the entrainment portion venturri is created and this causes the total flow exiting the nozzle to be several times greater than the input fluid volume supplied. Fig. 4.7 shows the cross sectional view of the nozzle that describes the nozzle region, entrainment region and inlet region more clearly. The flow from the nozzle exit divided by the inlet flow into the jet is referred as the *magnification ratio*. In this research the focus is to achieve optimum nozzle geometry to achieve the maximum magnification ratio.

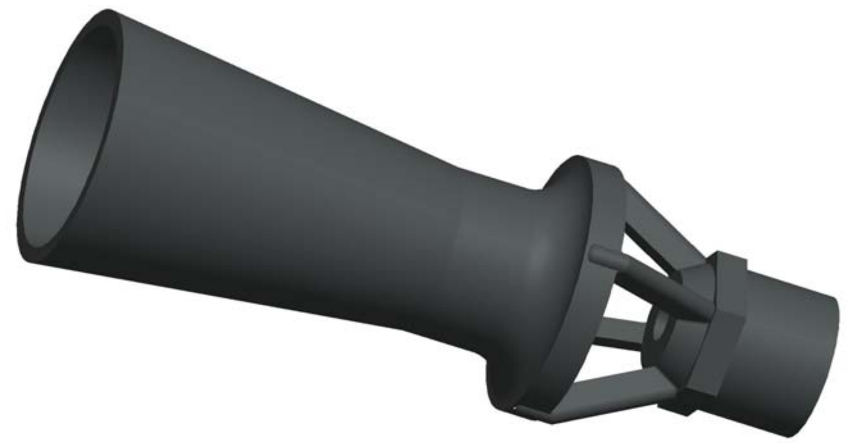


Fig. 4.6 CAD Model of a Hydraulic Mixing Nozzle

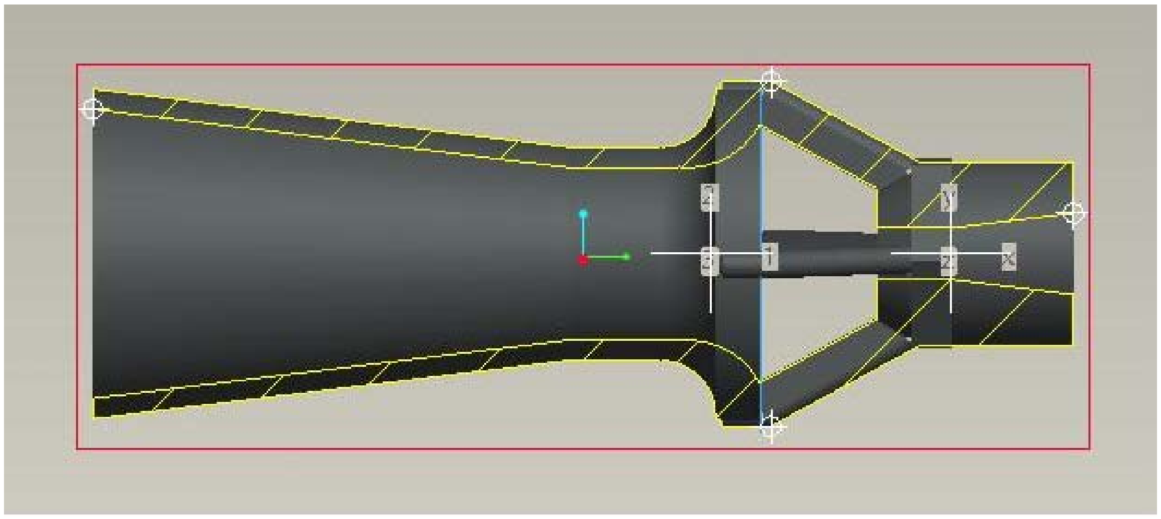


Fig. 4.7 Simple Cross-sectional View of Hydraulic Nozzle

Despite the availability of several off-the-shelf nozzles of this type that are optimized to operate near a specific flow rate and pressure conditions, the goal of this work is to develop the optimum nozzle to better suit the needs of this problem. The operating conditions include the jet diameter of 5/16” and the jet exit speed of approximately 60 ft/s. The magnification ratio of the nozzle design depends on the nozzle length (horn area), entrainment length (open area), and the horn profiles. Thus the design variables of interest are nozzle length, entrainment length and the number of horn points and its location. This is an interesting flow problem involves fluid mixing. The optimum nozzle is obtained when the evolutionary optimization technique is integrated with a CFD solver. More details regarding this problem can be referred from Engelbrecht et al., [To be Submitted]; Xiao et al., [2005]. The goal of this research work is to study the impact of progressive designer interaction on expediting the nozzle optimization. CFD solver coupled with an optimization algorithm running for a few weeks is not uncommon. In this research, the positive steps are taken to expedite the nozzle optimization by reducing the fruitless calls to the solver i.e., the designer intelligence are used to clear the low performing solutions.

4.3.1 Evolutionary algorithms for optimum nozzles

Engelbrecht et al., [To be submitted] studied the use of graph based evolutionary algorithms coupled with CFD to seek an optimum nozzle design. In their study, four different combinatorial graphs each with varying levels of graph connectivity were used to explore optimum designs of the nozzle. This work is built upon the work of Engelbrecht et al. Simple evolutionary algorithms are used in place of graph based evolutionary algorithms, because the EAs are the part of the progressive interaction system developed in this work. EA coupled with the Star-CD is used in this research to achieve nozzle optimization. Multiple possible solutions exist and wide search space exploration is desired, hence the evolutionary optimization techniques are preferred. Within the regional constraints presented by the designer on nozzle length, entrainment length and the horn point location, a population of nozzles can be represented using evolutionary algorithms. Each nozzle describes a specific location in the search space thereby the population diversity is preserved. Thus a large number of nozzle designs can be evaluated during the conceptual design phase, which is a requirement. From the pool of good nozzle, the best performing nozzle can be chosen.

4.3.2 Description of nozzle setup

The nozzle used in this work was injection molded from a non-corrosive polypropylene plastic. The orifice diameter of the nozzle is (5/16”), the entrainment length is (0.7”), and the nozzle horn length is 3.9”. For optimization purposes, the nozzles with the horn lengths regions ranging from 3 to 8 inches were investigated. The number of horn points that describe the nozzle profile was taken to be 50. The acceptable range of entrainment (or open area) length was 0.2 to 4.0 inches. A simple two dimensional model was used for the CFD analysis. The Z axis is along the breadth of this paper, the X axis is along the length of this paper. The nozzle model shown in Fig. 4.7 is along the X-Z plane. Due to the symmetry only one half of the model shown in Fig. 4.7 is used for CFD analysis. The horn points in the X axes were in the range of 0.5 to 1.5 inches and that in the Z axes were allowed to be between 0 to 18 inches. The rightmost of Fig. 4.7 is the origin. The inlet length (jet region) was maintained to be constant (5"), and the inlet diameter was 0.5". The magnification ratio of the nozzle is defined by equation (4.9).

$$Mag = \frac{f_{out}}{f_{jet}} \tag{4.9}$$

CHAPTER 5. PROGRESSIVE DESIGNER INTERACTION

The goal of this research is to develop and implement evolutionary optimization techniques coupled with high-fidelity models in an interactive environment. All the activities (interaction, visualization and computation) are controlled by the designer-in-the-loop. Progressive interaction ensures thorough exploration of the search space and hence, the quality of the optimum solutions. A flexible software framework is needed to implement the progressive interaction scheme. This chapter presents the implementation details human-guided, progressive designer interaction scheme using VE-Suite. The list of the attributes of progressive interaction system is presented in Section 5.5.

5.1 VE-Suite

VE-Suite (www.vesuite.org) is an open source virtual engineering software package that is currently under active development by the Virtual Engineering Research Group at Iowa State University. VE-Suite is designed as a high-level support tool for engineers who want to transform their traditional applications into virtual engineering-based applications. Essentially, VE-Suite enables users to easily incorporate component models and corresponding two-dimensional and three-dimensional graphical representations to create new, plug-and-play framework components. The goal is to enable engineers to carry out geometric modeling, performance analysis, and numerical analysis *en route* to engineering design within a virtual environment. In addition, virtual engineering technology serves as a means of gaining insight into the design space. The modular development of VE-Suite makes it more flexible to handle most engineering decision making problems that involve creating and analysis of high fidelity models, optimization, and decision making. Every time a new project is introduced to this framework they are done as a "plug-and-play" addition, without any modification to this framework. This chapter provides the details on how VE-Suite is used to create a progressive interaction environment for optimization.

5.2 VE-Suite Structure

The framework of VE-Suite is shown in Figure 5.1. The core modules of VE-Suite are VE-Xplorer (the graphical engine which is used to view comprehensive two-dimensional or three-dimensional graphic results), VE Conductor (the GUI front end to the virtual engineering framework which provides easy user interaction), and VE-CE (the computational engine). VE-Suite is general in nature and the three key components can run separately on a geographically diverse set of heterogeneous computer platforms. This separation is convenient because the VE-CE can run on the same machine as the application (computational unit), and VE-Conductor, which presents a graphical user interface to the user, can execute remotely on a separate machine. For example, the VE-CE component can run on a Linux cluster; the VE-Xplorer component can run on an SGI rendering machine; and VE-Conductor can run on a portable Tablet PC. Therefore, the framework components can be distributed across computational resources to make the most efficient use of these resources. This architecture is also advantageous because VE-CE must exist through the application's lifetime while VE-Conductor does not share this requirement. VE-Conductor is transitory and can connect to the server many times throughout the server's (application's) existence. Since the client may use visualizations for data interpretation, the end-user may choose to run the client on a high-performance graphics system. Also, the three core components of VE-Suite can function as complete stand-alone applications provided the necessary input files are prepared by the user.

The communication between the different components and user-defined modules is built upon the widely adapted and stable Common Object Request Broker Architecture (CORBA) standard developed by the Object Management Group over the last decade. The Executive module (one of the key modules in VE-CE) implements two of the standard CORBA services bundled with The Ace ORB (TAO) CORBA [http://www.ece.uci.edu./~schmidt/TAO.html] distribution. The first service is the COSS Naming Service that is used as a lookup table of currently running processes which allows clients to find running process based on a given ID. The second service is the interface that houses the functional and data type definitions and provides them graphically to the user to allow him or her to define a workflow in the graphical interface. An Interface Definition Language (IDL) between VE-CE and other components was designed to generate general data types in order to meet the requirements of different applications.

In VE-Suite framework, the running process (usually the computational unit) can broadcast its status and analyze information from multiple GUI clients. Any given GUI can connect to the system information stream at any point in time and view the current state of the running process. The framework is designed to allow the GUI to be shutdown and restarted at will without any impact on the computational unit's execution. This attach/detach functionality gives the user the ability to easily monitor the computational process. As an example, this functionality allows a user to build and start a simulation and then detach from the computational engine. The user could then go to a different location, re-attach to the running process, and regain monitoring and control functions. The other advantage of the use of component architecture design techniques is that multiple GUIs can also be connected simultaneously from different computers and allow multiple users to monitor a simulation from different locations. To enable portability on multiple operating systems and immersive technology platforms, VE-Xplorer is built upon VR Juggler [www.vrjuggler.org], Open Scene Graph [www.openscenegraph.com], and Kitware's Visualization Toolkit (VTK) [www.vtk.org].

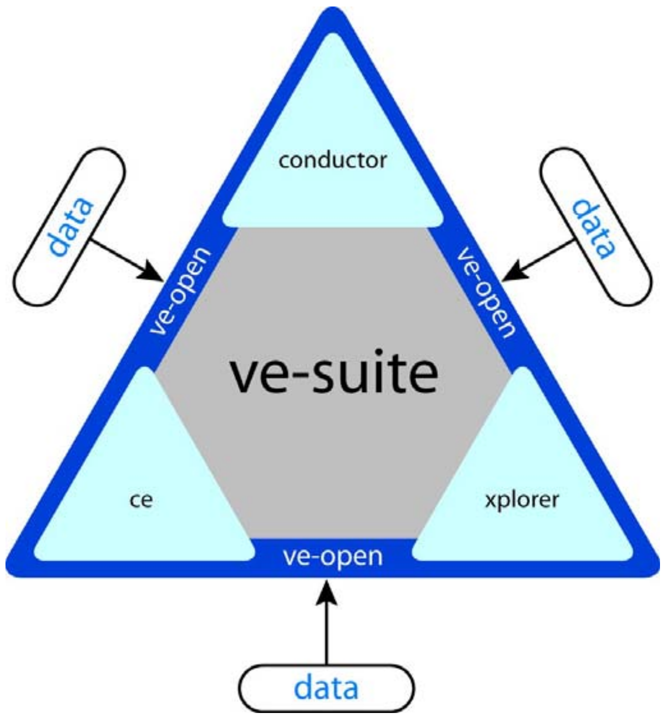


Fig 5.1 Architecture of VE-Suite

5.3 VE-Suite API

When building a virtual engineering-based application, the implementation can be broken-down into three groups: VE-Conductor for user interface, VE-CE for computations, and VE-Xplorer for the visualization of results. As shown in Fig. 5.1, these groups form the building blocks of the VE-Suite API.

5.3.1 VE-Conductor

The GUI is where the user is able to create the system configuration, set model inputs, start and stop execution of the simulation, and view simulation results based on the system configuration they designed in the first step. Developers can create their own GUIs and then compile the resulting code into a Dynamic Link Library (DLL) in Windows or a shared library in Linux/Unix. Thus, it is more likely to meet user specific requirements because the interface is built by the person who will use it. This user interface will be custom-tailored to the user's needs and can easily be modified if the user's focus of interest changes. VE-Suite is able to dynamically discover, identify, and load the user's GUI from these shared libraries. VE-Suite provides a simple API built on top of wxWidgets for the user to build his own interface to fit the requirement of the actual application. WxWidgets was chosen as the UI library because it is one of the most functional, stable cross-platform UI libraries.

5.3.2 VE-CE

As mentioned before, VE-CE is used to construct, coordinate, schedule, and monitor the running processes. VE-CE provides a CORBA server with which the detachable GUI and computational unit connect. It is capable of running simulations containing a multitude of different types of models, each accepting and generating a myriad of data types. Since users from different fields have a wide variety of needs, the analysis tools and their use require a detailed understanding of the problem. Therefore, it is the user's responsible to develop their application objects (computation unit). VE-Suite uses the concept of the computational unit to allow flexibility of the system. This decreases coupling between the GUI and computation unit. Changes to GUI or unit will not affect the other unless the change involves a change of the system configuration. Once a client-server connection is made, the GUI is able to send parameters and commands to the unit.

5.3.3 VE-Xplorer

A key aim of virtual engineering is to fully engage the human capacity for problem solving by creating a realistic experience for the user so that he or she can focus entirely on the engineering problem. The advantage is that previously indescribable complexities can be understood and the full range of engineering solutions can be explored. The graphical engine (VE-Xplorer) provides the core visualization functionality for the virtual engineering aspect of the framework. It can load geometry files, three-dimensional simulation data and experimental data of almost every format into a scene. VR Juggler is used to handle interfacing with VR hardware and graphics rendering platforms. VE-Suite handles the creation of the virtual environment and VR Juggler allows software to run with any type of virtual environment, from a regular 2-D screen to a six-walled immersive virtual space. Due to the generality of the visualization requirements, the VE-Suite core provides a complete visualization GUI so that users can navigate and control the scene. Thus, although most users of VE-Suite are not necessarily expert software developers, they need not worry about the complexities of details of graphics and virtual reality programming and can instead spend time on their applications. More details regarding the implementation of VE-Suite can be found in [Huang, 2006].

5.4 Requirements of Interactive Environment

The three most common modes of progressive interaction are inspection, modification, and designer-controlled re-optimization. Inspection is an interactive mode that requires some output from the interactive environment. This is the mode through which the interactive system communicates to the designer. Research study indicates that nearly one-half of the brains neurons are active for visual information [Lotov et al., 2004]. Hence, the most common and powerful mode of inspection is graphical visualization. Modification is the mode through which the designer communicates their preferences to the interactive system. This is typically done by graphical user interface (GUI). The ability to control the inspection and re-optimization rests upon modification. Designer controlled re-optimization does not directly communicate with the designer. Rather the results obtained from this process are converted to the suitable form that is readily understandable by the designer. All the three modes of progressive interaction can be harmonized if the appropriate division of labor between the designer and computers are achieved. The humans' superior abstract thinking and the computers' superior computational speed can work together to produce a synergistic effect. Having described the functionality of each interaction mode, and the expectations from the interactive environment, the next step is to consider the basic requirements of the interactive environment. Any interactive environment should meet the following requirements:

1) Simplicity: The Application Programming Interfaces (APIs) and/or languages used to create applications should be designed simply so that the designer can understand without much delay. This means users from different fields should be able to easily build applications

inside the system or add new capabilities without dealing with system programming issues. The implementation should hide as much of the system's underlying complexity as possible.

2) Extensibility: The system should be easy to extend the existing capabilities. This requirement is highly desired as the design changes, project changes may result in change of product configuration completely. The current code should be capable of accepting the changes without major alterations.

3) Flexibility: The system should enable users to choose from a variety of engineering tools in a platform independent manner.

One of the major requirements of a progressive interaction design environment is to provide users with sufficient data and other information about the designed product and its performance in an intuitive manner. VE-Suite provides such functionalities, such as a detachable GUI, realistic graphical engine and bi-directional bindings between the GUI and the computational unit. This work focuses on developing specific applications built upon the interactive evolutionary design environment. For instance, the basic framework of VE-Suite and data handling structure of interactive evolutionary design environment remains the same. Specific implementation done on GUI, VE-CE, and graphical engine to suit progressive interaction is presented in this chapter and specific implementation are presented with individual case study description.

5.5 Implementation of Progressive Interaction

Progressive interaction is defined as a human-guided preference articulation method where the designers' interaction continuously controls the engineering optimization by visualization, modification and controlled re-optimization. In progressive interaction designer preference articulation is done continuously *during* the optimization process. To achieve real-time or near real-time response the communication between the user interface, computational unit, and visual graphical module.

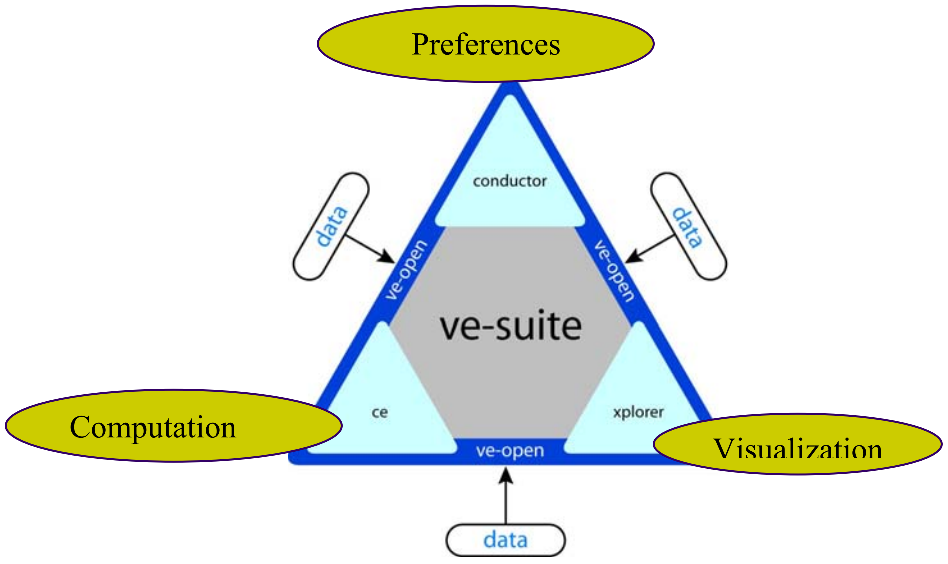


Fig. 5.2 Progressive Interaction Using VE-Suite

This section focuses on describing how the progressive interaction for engineering optimization is built within the VE-Suite software framework. The modular configuration of VE-Suite extends well to accommodate the basic modules of the progressive interaction system, as shown in Fig. 5.2. The three basic modules that need to be integrated to implement progressive interaction are a preference articulation module, a computational module, and a visualization module. All three modules are controlled by the designer. Human interaction with visual feedback on the state of the problem helps designers change parameters during interaction.

5.5.1 Parameters for interactive optimization

VE-Conductor is the user interface used in the VE-Suite framework with standard features to control the graphical engine. The low-level (or optimization method specific) parameters and high-level (or problem specific) parameters are to be incorporated into the user interface through the graphical user interface (GUI) plugin. The parameters whose values need to be transferred to the computational unit have to be registered. The code fragment that accomplishes this task for the finned dissipater problem is shown in Fig. 5.3.

```
HeatExchanger::HeatExchanger( )
{
        name = "HeatExchanger";
        RegistVar("mutationrate", &mutationrate);
        RegistVar("crossoverrate", &crossoverrate);
        RegistVar("popsize", &popsize);
        RegistVar("replacemethod", &replacemethod);
        RegistVar("crossovermethod", &crossovermethod);
        RegistVar("mutationmethod", &mutationmethod);
        RegistVar("checkinggene", &checkinggene);
        RegistVar("terminationmethod", &terminationmethod);
        RegistVar("designparamnames", &designparamnames);
        RegistVar("designparams", &designparams);
        RegistVar("upperlimits", &upperlimits);
        RegistVar("lowerlimits", &lowerlimits);
        RegistVar("activeindexs", &activeindexs);
        RegistVar("inactiveindexs", &inactiveindexs);
        RegistVar("checkingcreature", &checkingcreature);
        RegistVar("totalruns", &totalruns);
        RegistVar("totalmatingevents", &totalmatingevents);
        RegistVar("degree", &degree);
        RegistVar("interactiontype", &interactiontype);
}
```

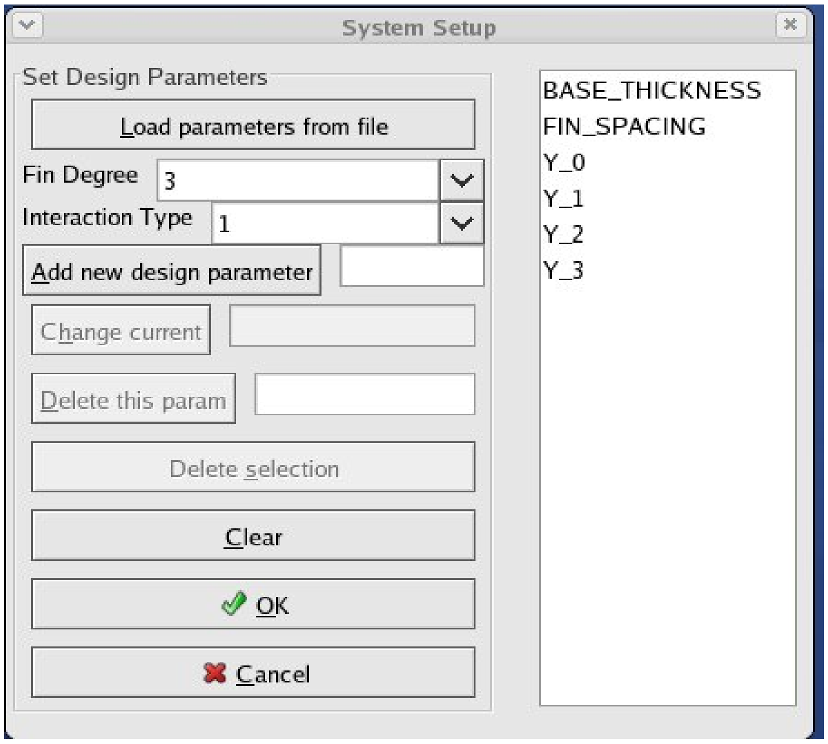
Fig. 5.3 Registering Parameters for GUI - Computational Unit Communication

High Level Parameters. These are problem specific parameters. Engineering optimization has objective function with a vector of decision variables, inequality and equality constraints. The constraints that are listed are explicit constraints, while the implicit constraints are to be discovered by the designer during the process of optimization. For an interactive optimization process, the designer should be able to alter the numerical bounds of the decision variables and to add and remove constraints “on-the-fly”. In this research work, the high level parameters of the problem are loaded to the GUI through a DesignSetup file. The numerical constraints for these high level parameters are also changed in the real time. For instance, in the image segmentation and optimization problem, the designer supplies the interactive system a digital image. Based on the size of the image file the numerical constraints of the design variables automatically changes. The DesignSetup and constraint file information for the image segmentation and optimization problem is presented in Fig. 5.4 for an aerial survey image whose size is 1188x844 pixels

X_GEN	0	1188
Y_GEN	0	844
WEIGHTS	0.0	100.0

Fig. 5.4 Design Setup and Constraints File Information

The DesignSetup file is generated in the run time based on user’s interest. For example, in the shape optimization of finned dissipater, the designer can choose the degree of the fin profile. Based on the degree value, the number of parameters required for the optimization changes. After inputting the required parameter values in the High-level parameters dialog, as shown in Fig. 5.5 the “Load Design Parameters from File” is chosen to dynamically vary the number of design parameters based on the fin degree. This change is dynamically updated in the DesignSetup file. The code fragment that accomplishes the dynamic updating of DesignSetup file is shown in Fig. 5.6.



Fin Degree: 3
Interaction Type: 1

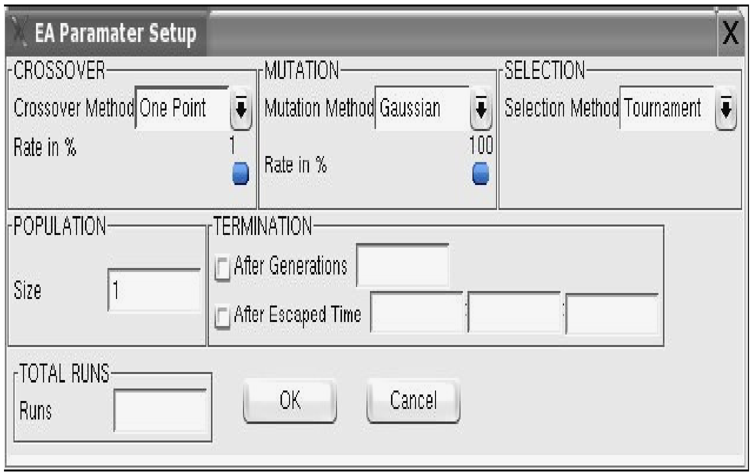
BASE_THICKNESS
FIN_SPACING
Y_0
Y_1
Y_2
Y_3

Fig. 5.5 High-level Parameter Setup Dialog

```
std::ofstream designsetup;
designsetup.open("DesignSetup.txt", std::ios::out);
designsetup<<"BASE_THICKNESS"<<std::endl;
designsetup<<"FIN_SPACING"<<std::endl;
for(int i=0; i<(degree+1); ++i)
{
        designsetup<<"Y_"<<i<<std::endl;
}
designsetup.close();
```

Fig. 5.6 Design Setup File Generator

Low level parameters. These are the solution method-specific parameters. These parameters are subjected to the designer interaction to expedite the optimization process. Typical low level parameters include population size, total mating events, total number of evolutionary runs, gene length of the solution, choice of parent and co-parent selection, crossover, mutation and replacement operators, as shown in Fig. 5.7.

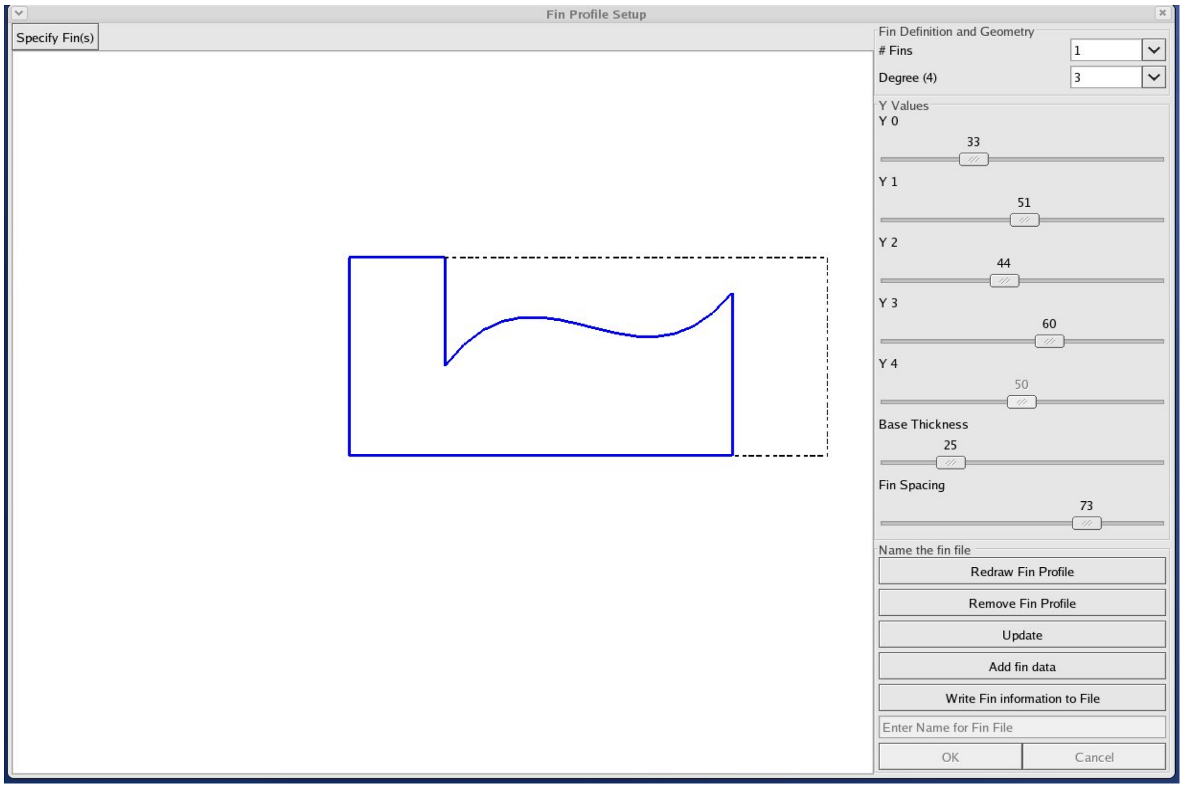


Crossover Method One Point
Rate in % 1
Mutation Method Gaussian
Rate in % 100
Selection Method Tournament
Size 1
After Generations
After Escaped Time
Runs

Fig. 5.7 Low-level Parameter Setup Dialog

In all the projects done as a part of this research work, the results of the computation from the computational unit are available to the designer either as numerical data or visual outputs in the graphical plugin.

Advanced human-guided interaction. This is achieved when the designer intelligence is used to develop the complete gene sequence for a few solutions in a population. For instance, in the shape optimization problem, a 32 member EA population was used for each fin degree (0 through 4). OpenGL based interactive design canvas was developed as a part of progressive interaction research is shown in Fig. 5.8. This design canvas generates the visual output of the fin profile developed based on the designer values. The user has the flexibility to select the most appropriate fin profile values very intuitively. These fin profiles generated based on the user input can be used as special members in the evolving population to direct the EA search.



Fins 1
Degree (4) 3
Y 0 33
Y 1 51
Y 2 44
Y 3 60
Y 4 50
Base Thickness 25
Fin Spacing 73

Fig. 5.8 Interactive Fin Profile Setup Design Canvas

Say in a 32 solution population, a maximum of 5 solutions in the population were generated by the designer. And these solutions were intended to guide the optimization process as elite members. As the population evolves, the designer can pick and mutate a solution to extend the exploratory capability of the solution. This advanced human-guided interaction is done without disturbing the optimization process. Also during the optimization run, the designer has the flexibility to completely replace few solutions from the population with the designer supplied solutions. This is done without disturbing the EA run. Generally, the solutions replaced are typically the less productive ones. This not only expedites the EA process, but also, accomplishes the intuitive exploration of the solution space with “what-if” analysis.

5.5.2 Computational engine

Low and high-level parameters including the choice of optimization technique, and interaction type are to be submitted to the VE-CE from the VE-Conductor. To start the computational engine the parameter values supplied by the user are required. The computational engine ties-up with the solvers, optimization routines, etc. The results of the optimization are continuously monitored by the designer. If the designer doesn’t find any improvement in the fitness function, then the designer understands that the algorithm is stuck in a local optimum. This problem can be sorted out by “injecting” new members in the EA population. Through the user interface a new set of parameters are supplied to the computational engine. After the optimization is done, the feasibility of the solution needs to be ensured this is done by visualization.

5.5.3 Interactive visual analysis

The results of the designer interaction are sent back to the designer in the form of numerical outputs. The designer interaction as mentioned above should result in the increase of fitness. The fitness progress report displays the current fitness value of the best solution, and the average fitness values. This will help designer evaluate the influence of their interaction on the optimization. The graphical engine of the VE-Suite displays the results of the optimization as supplied by the computational unit. The contents of the graphical engine are also controlled by the GUI. For instance, in the fin shape optimization problem, the computational unit optimizes the fin profile and the visualization toolkit (VTK) files are generated for the optimized profile. The designer can chose any VTK file that was output in the intermediate stages of the evolution and load it in the graphical engine. The output from the graphical engine for an intermediate VTK file is shown in Fig. 5.9.

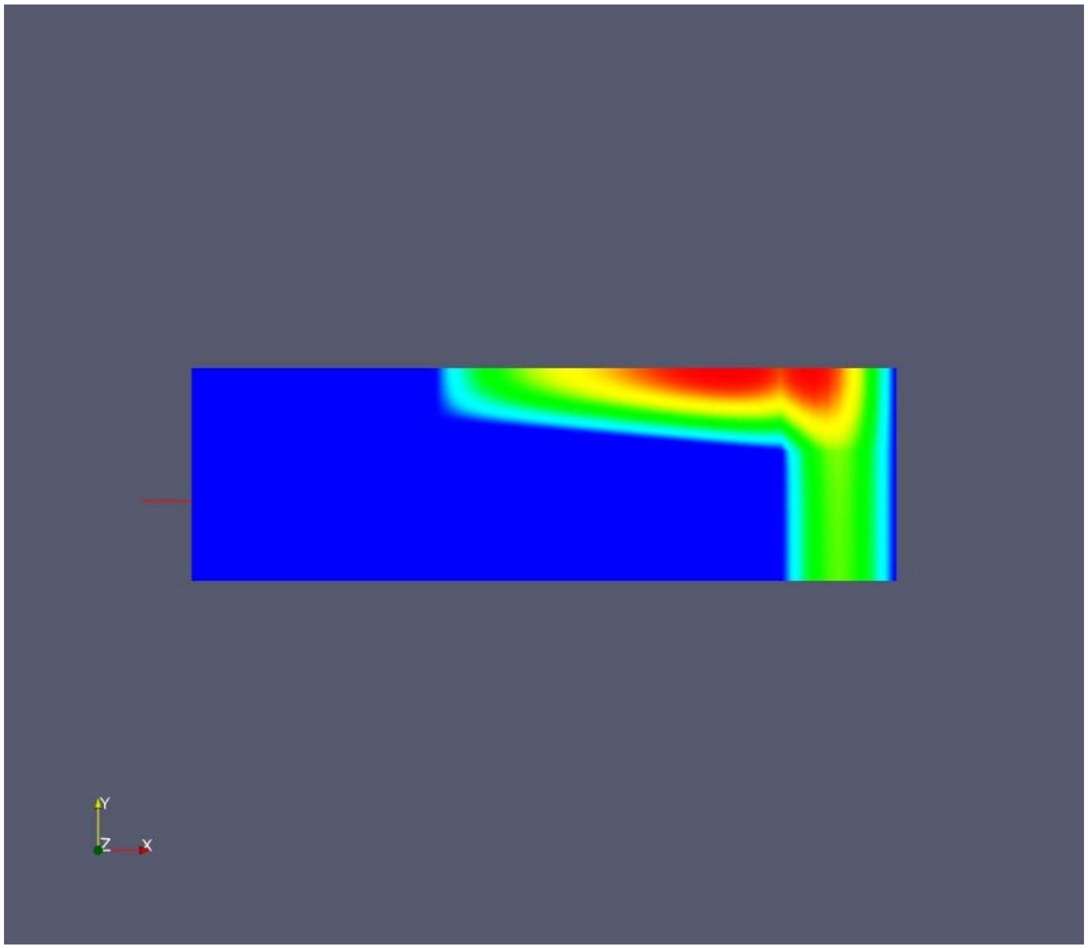


Fig. 5.9 Visual Output of Velocity Distribution in a Third Degree Fin Profile

If the designer finds the intermediate profiles more interesting, then the EA runs can be controlled by injecting more suitable profiles into the population or terminating the runs. Also, in the graphical engine, the resulting optimum fin profile can be visualized with the temperature and velocity distribution. If the designer understands the optimized profile is infeasible for implementation then a set of problem-specific parameters are altered and optimized again, this completes the loop of human-guided visual analysis.

5.6 Contribution of this Work to VE-Suite

The work presented in this chapter is added to the VE-Suite framework as external plugins. This is applicable to all the three test cases studied as a part of this research work. Certain unique contributions made as a part of this work which can be integrated in the core VE-Suite code in future is presented in this section.

5.6.1 Inputs to VE-Conductor

The interactive design canvas class was developed using the wxWidgets and OpenGL software. This interactive design canvas is integrated with the designer preference module. This canvas can operate as a start-up to the optimization process, during the optimization process and after the optimization runs. Firstly, as a start-up this canvas is used to prepare the designer choice of solutions for the evolutionary optimization. The real-time visual display of the designers' preference enables the designer to decide whether to proceed with that particular solution or not. Secondly, during the evolutionary optimization runs are in progress, the designer studies the progress of the EAs from fitness output files. If the designer senses the stagnation of the EA, then the interactive design canvas is used to generate a set of solutions and insert those in the population.

As a part of this work, image segmentation and optimization is introduced as a plugin to VE-Suite. Once compiled, this plugin can handle any digital image, based on the image size the constraints required for image segmentation and optimization get reassigned. Based on whatever image the designer is desired to segment, the interactive canvas class can load the image in the design canvas and accept the user input of Voronoi generators and their corresponding weights. The designer can opt to choose whether to insert solutions during the runtime or not. In the shape optimization of the finned dissipater test case the designer can use the same designer preference module for any degree. Based on the degree the design variables can change on-the-fly and their associated constraints do also change. In the hydraulic mixing nozzle problem, the designer can pre-specify the number of horn points and generate nozzle profiles, and these profiles are supplied to the computation unit at the runtime. In addition to the nozzle profiles, a whole range of solver parameters, flow parameters, mesh parameters are all supplied from the designer preference module. All the contributions done for this research work are developed in modular basis and can readily be integrated with the VE-Suite directly or as a dynamic linking library (DLL).

5.6.2 Inputs to VE-CE

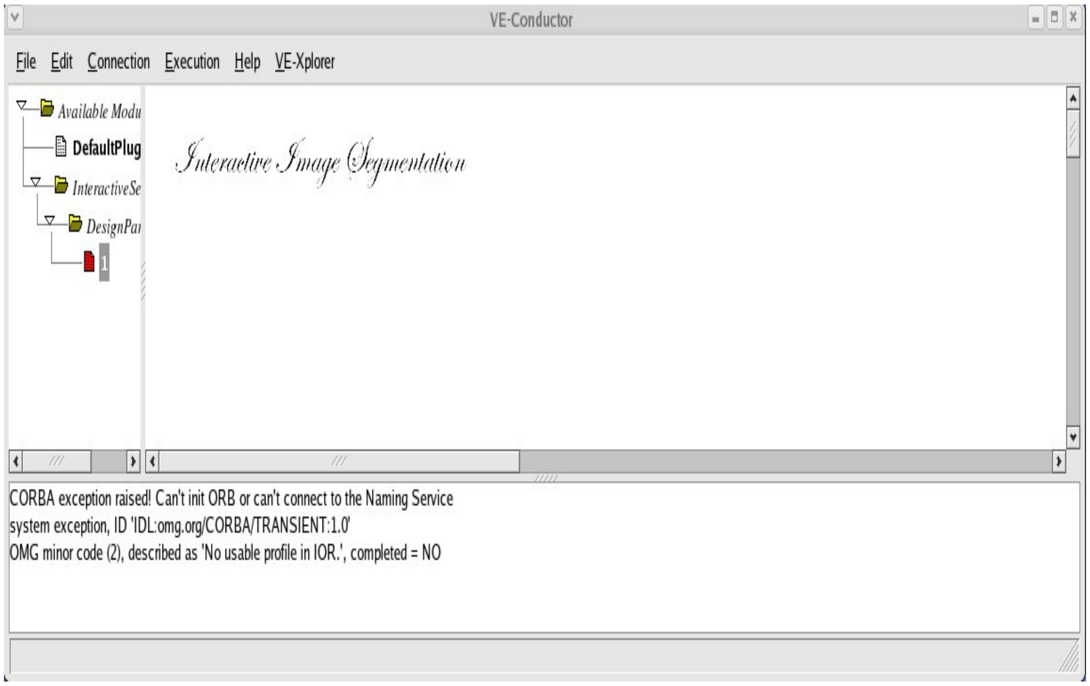
The computational units received designer preferences initially. During the optimization runs, the computational units are controlled directly by the designer who gives the information on the parent selection for the evolutionary mating events. Designer also can insert a new member in the population during evolutionary run and replace the least performing member without disturbing the optimization runs. After every ten mating events, visual image output is generated for both image segmentation and shape optimization test cases. Due to the complexity and high time consumption, the image outputs for the best nozzle are done once after an evolutionary run is completed. As an enhancement to this work, multiple computational units, say for instance computational unit that handles CFD and CAD portions of the nozzle can be seamlessly tied. For example, the CAD and CFD modules, where the CAD portion focuses on accepting the user information on the horn points and builds the three dimensional model of the part. While, the CFD nozzle can be set to conduct optimization runs and the best nozzle profiles information is resubmitted to the CAD. This enhancement demonstrates the capability of progressive designer interaction system to handle multiple engineering tools in a virtual engineering environment to facilitate engineering design.

CHAPTER 6. IMPLEMENTATION FOR IMAGE SEGMENTATION AND OPTIMIZATION

Low-impact segmentation represents almost invariant image segments using a single color so that the quality degradation is minimal. The larger the number of segments, the more closely the segmented image replicates the original image, but the higher the computation time. Representing the image with an optimal number of segments minimizes the file size while maintaining acceptable perceptual image quality. But determining the optimal number of image segments is a non-trivial task. This results in an interesting optimization problem with the existence of multiple feasible solutions. This necessitates the development of human-guided, interactive image segment generation and optimization tool. Designer interaction helps in generating optimal image segments with an acceptable image quality to make engineering decisions. The designer is directly involved in identifying the interesting areas of the image. For example, the designer may choose to have more segments in information rich areas and vice versa. The selection of interesting areas of the image directs the optimization to fruitful areas of the search space. This section presents the implementation details of human-guided, progressive designer interaction tool for image segmentation and optimization.

6.1 Designer Interface Module

This module is used by the designer to input their preferences, low-level, and high-level parameters required for the optimization. This module is introduced as the graphical user interface (GUI) plugin within the VE-Suite. The main page of the designer preference module is shown in Fig. 6.1.


VE-Conductor
File Edit Connection Execution Help VE-Xplorer
Available Modu
DefaultPlug
InteractiveSe
DesignPar
1
Interactive Image Segmentation

```
CORBA exception raised! Can't init ORB or can't connect to the Naming Service
system exception, ID 'IDL:omg.org/CORBA/TRANSIENT:1.0'
OMG minor code (2), described as 'No usable profile in IOR.', completed = NO
```

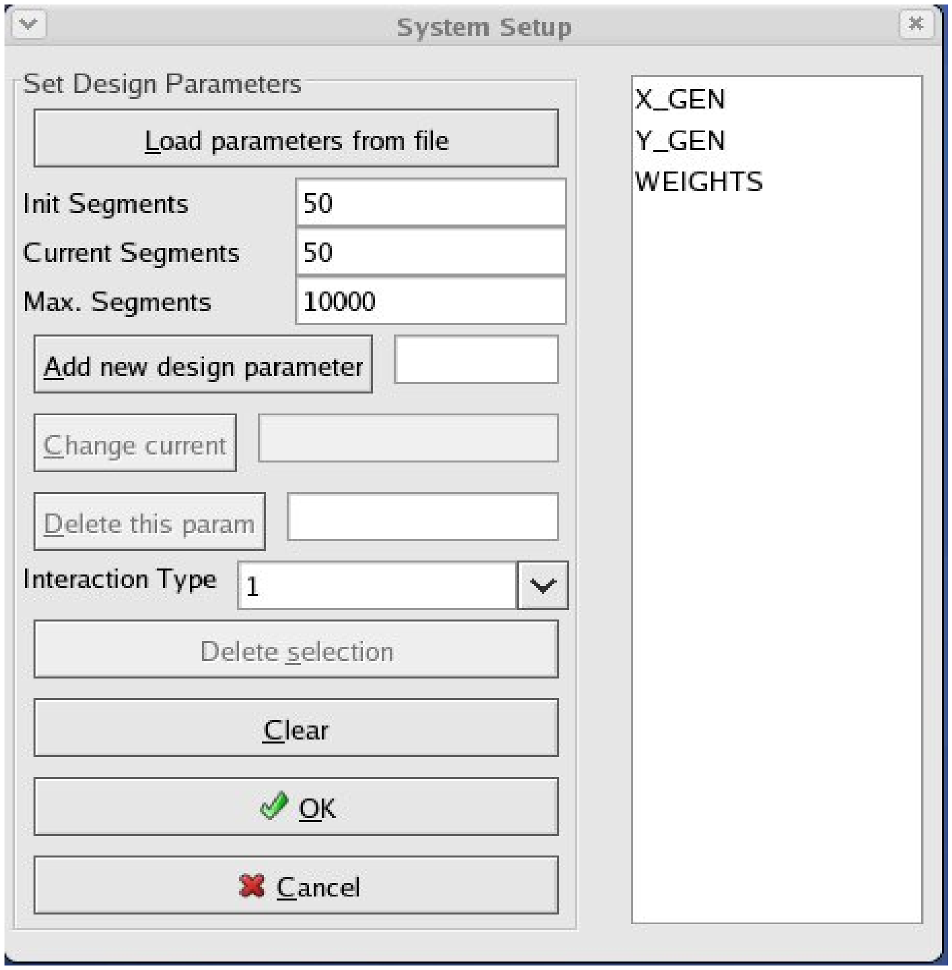
Fig 6.1 Design Preference Module – Main Window

The low-level and high-level parameters (or variables) that need to be transferred to the computational unit have to be registered. The code fragment that registers the variables for the interactive image segmentation project is presented in Fig. 6.2. The user interface module is generic, and pre-compiled implementation, i.e., the designer can work on their choice of image without having to compile the code. The designer interaction plugin begins with the input of a Portable Pixmap (ppm) image file. The ppm file consists of pixel location and its color values in the ASCII format. The image file is input using regular file input dialog. In the next step, the designer needs to provide the interactive system with high-level parameters.

```
InteractiveSegmentation::InteractiveSegmentation( )
{
            name = "InteractiveSegmentation";
            RegistVar("mutationrate", &mutationrate);
            RegistVar("crossoverrate", &crossoverrate);
            RegistVar("popsize", &popsize);
            RegistVar("initsegments", &initsegments);
             RegistVar("currsegments", &currsegments);
            RegistVar("maxsegments", &maxsegments);
            RegistVar("combgraphname", &combgraphname);
            RegistVar("replacemethod", &replacemethod);
            RegistVar("crossovermethod", &crossovermethod);
            RegistVar("mutationmethod", &mutationmethod);
             RegistVar("checkinggene", &checkinggene);
            RegistVar("terminationmethod", &terminationmethod);
             RegistVar("designparamnames", &designparamnames);
             RegistVar("designparams", &designparams);
             RegistVar("upperlimits", &upperlimits);
            RegistVar("lowerlimits", &lowerlimits);
            RegistVar("activeindexs", &activeindexs);
            RegistVar("inactiveindexs", &inactiveindexs);
             RegistVar("checkingcreature", &checkingcreature);
             RegistVar("totalruns", &totalruns);
            RegistVar("totalmatingevents", &totalmatingevents);
             RegistVar("stdimagefilename", &stdimagefilename);
            RegistVar("interactiontype", &interactiontype);
}
```

Fig. 6.2 Variable Registration Code

High-level parameters setup. The high-level parameter setup or the design parameter setup user interface is illustrated in Figure 6.3. The X_GEN, Y_GEN and WEIGHTS in the right side of the design parameter setup user interface indicate the high level parameters. The X_GEN and Y_GEN correspond to the tile centers and WEIGHTS represent numerical weights of the tile centers. The number of design parameters remains the same for all the test images. Based on the input image file and the design parameters a constraint file is automatically generated.



Init Segments 50
Current Segments 50
Max. Segments 10000
Interaction Type 1

X_GEN
Y_GEN
WEIGHTS

Figure 6.3 High-level Parameter Setup User Interface

The typical constraint file for an 1187x844 aerial survey image is shown in Fig. 6.4. The code fragment that accepts the designer supplied input image file and automatically generates the constraints file is shown in Fig. 6.5.

```
  X_GEN       0     1187
   Y_GEN       0     844
 WEIGHTS     5.0   100.0
```

Fig 6.4 Design Constraints for an Aerial Survey Image

```
Void UI_TopFrame::CreateConstraintFile
{   wxFileDialog dialog
                ( this,
                  _T("Load image file"),
                  _T(""),
                  _T(""),
                  _T("All files (*.*)|*.*")
                );

    dialog.SetDirectory(wxGetHomeDir());

    if (dialog.ShowModal() == wxID_OK)
    {
      wxString info;
      std::string stdinfo;
      info = dialog.GetPath().c_str();
      ReadImageFile(info);
      stdinfo = info.mb_str();
      SetImageFileName(stdinfo);
    }

    unsigned int width, height;
    height = GetImageHeight( );
    width =  GetImageWidth( );

    std::ofstream constraintsetup;
    constraintsetup.open("constraints.txt", std::ios::out);
    constraintsetup<<"X_GEN"<<"\t"<<"0"<<"\t"<< width << std::endl;
    constraintsetup<<"Y_GEN"<<"\t"<<"0"<<"\t"<< height << std::endl;
    constraintsetup<<"WEIGHTS"<<"\t"<<"5.0"<<"\t"<<"100.0"<<std::endl;
    constraintsetup.close();
}
```

Fig. 6.5 Input Image Setup and Constraint File Generation

The initial image segments, the current image segments, and the maximum image segments are the parameters that determine the starting conditions and stopping criteria for the segment optimization process. The interaction type can be chosen as either 0 (random) or 1 (interactive). When the interaction type is chosen to be 1 the optimization process can be controlled on-the-fly by using designer generated image segment locations. The designer created interactive population files get loaded. And the designer gets the opportunity to develop the tile centers using an interactive segment generator canvas according to their knowledge or point of interest in the image. The interactive segment generation can be done even when the optimization runs are in progress.

Advanced interaction - interactive segment generator canvas. The interactive segment generator canvas was specifically developed for this research work. This canvas class shown in Fig. 6.6 was developed using OpenGL and wxWidgets. The OpenGL takes care of the drawing or removing images and tile centers (or points) in the visual area. The wxWidgets was used for user interaction and file generation purposes. A brief tour of this canvas class is presented in this section. On the top layer of this canvas “Load Icon”, “Init Generator Points”, and “Interactive Generator Points” buttons are listed. The “Load Icon” initiates the input image file, in Fig. 6.6 the aerial survey image is loaded. The number of image segments (or points) needs to be input. Using this canvas the designer can choose from 50 up to 2800 segments. After the number of segments was chosen, the “Init Gen. Points” button needs to be selected to get the initial (random) tile centers in the screen. This is represented by small black dots on the screen. Once the tile centers show up on the screen, the designer can interactively (by mouse) move these tile centers to very interesting (high feature density) regions of the image. After the tile centers are moved to the appropriate locations (as desired by the designer) then the “Interactive Gen. Points” or “Redraw Points” button should be clicked. The screen will display the latest tile center locations. “Write Points Data” should now be selected to enable the designer enter the filename to store the tile centers and their corresponding weights. This file now holds a solution for the evolutionary optimization.

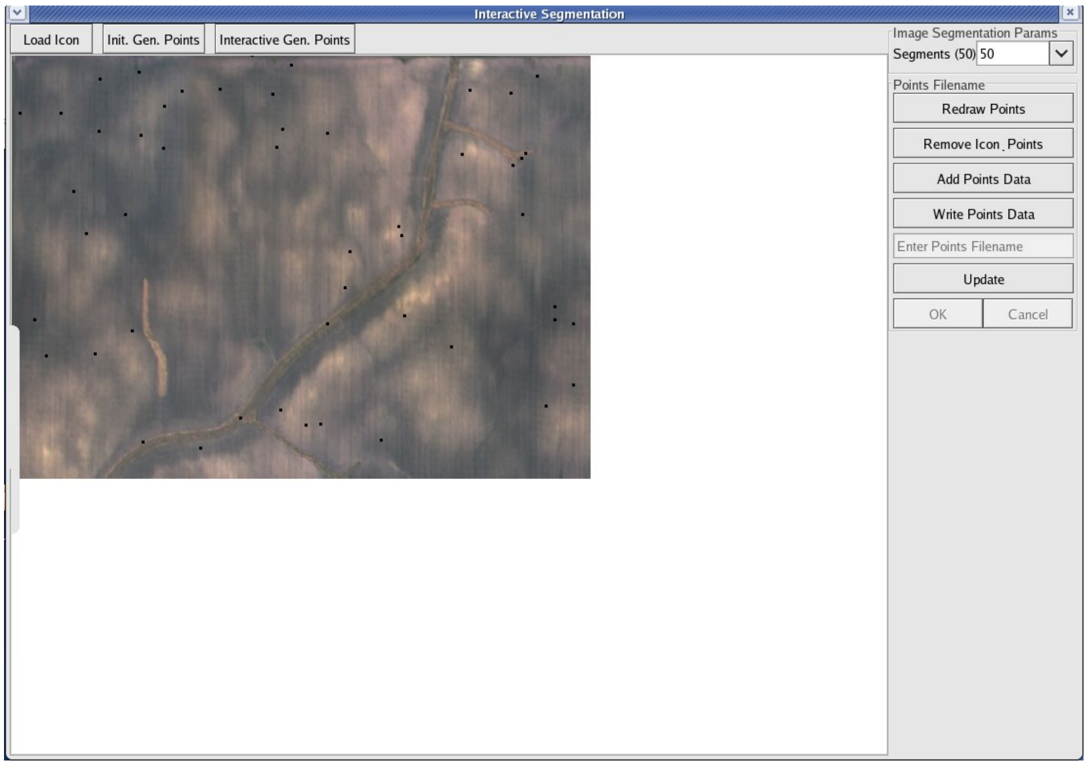


Figure 6.6 Interactive Image Segmentation Canvas

If multiple interactive solutions are to be generated, then the same procedure listed above should be followed and "Add Points Data" button has to be clicked to append to the tile center data to the same file. Using "Add Points Data" any number of interactive solutions can be generated. "Remove Icon & Points" button should be clicked to clean up the canvas window to get clear white screen. This interactive segmentation canvas is exited by selecting "Update" button.

Low-level parameters user interface. The low-level parameters (method specific) such as population size (32), number of mating events (200), number of evolutionary runs (10) is provided by the designer. The choice of mutation, crossover, and replacement are to be given to the system in addition to the stopping criteria for the evolutionary algorithms. The stopping critera for the evolutionary runs are generally designer specified, they are, (i) if the number of segments of a solution reaches the maximum number of segments specified by the designer, or (ii) if the specified maximum number of mating events are reached, or (iii) if image specific minimum segment variation is achieved. The designer inputs the low-level parameter using the user interface shown in Fig. 5.7.

6.2 Interactive Optimization Module

This module is developed as a plugin to the computational engine (VE-CE) of the VE-Suite. The variables registered in Fig. 5.4 are sent across to the interactive optimization module. Using VE_XML classes of the VE_Suite the both the high and low-level parameters, their corresponding numerical limits are sent to the optimization module. The interaction designer need not be an expert in XML as the VE_XML implementation of VE-Suite completely takes care of data transfer. The optimization module code fragment that accepts these parameters are shown in Fig. 6.7. The values accepted by the computational unit are “set” and “get” using parameter class. The parameter class methods are accessible by almost all the classes of the computational unit.

```
void Body_Unit_i::SetParams(::CORBA::Long module_id, const char*
param)
ACE_THROW_SPEC((CORBA::SystemException,Error::EUnknown))
{

      std::cout<<UnitName_<<" :SetParams called"<<std::endl;

      unsigned int i,j;
      VE_XML::XMLReaderWriter networkWriter;
      networkWriter.UseStandaloneDOMDocumentManager();
      networkWriter.ReadFromString();
      networkWriter.ReadXMLData(param,"Command","vecommand");
      std::vector<VE_XML::XMLObject*>
      objectVector=networkWriter.GetLoadedXMLObjects();
      std::cout<<"Object vector
      size:"<<objectVector.size()<<std::endl;

      // This displays all the params (in XML format) transferred
      from GUI to Unit

      std::string commandName;
      callcount++;
      if(callcount==1)
      {
         for (i=0; i<objectVector.size(); i++)
         {
            VE_XML::Command* params =
                         VE_XML::Command* >(objectVector.at( i ) );
            unsigned int num = params->GetNumberOfDataValuePairs();
            commandName = params->GetCommandName( );
            VE_XML::DataValuePair* curPair;

            if( commandName == "popsize" )
            {
               curPair = params->GetDataValuePair("popsize");
               curPair->GetData( popsize );
            }
            else if( commandName == "initsegments" )
            {
               curPair = params->GetDataValuePair("initsegments");
               curPair->GetData( initsegments );
            }
            else if( commandName == "currsegments" )
            {
               curPair = params->GetDataValuePair("currsegments");
               curPair->GetData( currsegments );

            }
            else if( commandName == "maxsegments" )
            {
               curPair = params->GetDataValuePair("maxsegments");
               curPair->GetData( maxsegments );
            }
            else if(commandName == "combgraphname")
            {
               curPair = params->GetDataValuePair("combgraphname");
               curPair->GetData(combgraphname);
               std::cout<<"combgraph"<< combgraphname <<std::endl;
            }
            else if(commandName == "stdimagefilename")
            {
               curPair =
                  params->GetDataValuePair("stdimagefilename");
               curPair->GetData( stdimagefilename );
            }
            else if(commandName == "checkinggene" )
            {
               curPair = params->GetDataValuePair("checkinggene");
               curPair->GetData ( checkinggene );
            }
            else if(commandName == "crossoverrate" )
            {
               curPair = params->GetDataValuePair("crossoverrate");
               curPair->GetData ( crossoverrate );
            }
            else if(commandName == "mutationrate")
            {
               curPair = params->GetDataValuePair("mutationrate");
               curPair->GetData( mutationrate );
            }
            else if(commandName == "replacemethod")
            {
               curPair = params->GetDataValuePair("replacemethod");
               curPair->GetDataString( );
            }
            else if(commandName == "crossovermethod")
            {
               curPair = params->GetDataValuePair("crossovermethod");
               curPair->GetDataString( );
            }
            else if(commandName == "mutationmethod")
            {
               curPair = params->GetDataValuePair("mutationmethod");
               curPair->GetDataString( );
            }
            else if(commandName == "terminationmethod")
            {
               curPair =
                  params->GetDataValuePair("terminationmethod");
               curPair->GetDataString( );
            }
            else if ( commandName == "designparams" )
            {
               curPair = params->GetDataValuePair("designparams");
               curPair->GetData( designparams );
            }
            else if ( commandName == "designparamnames" )
            {
               curPair =
                  params->GetDataValuePair( "designparamnames");
               curPair->GetData( designparamnames);
            }
            else if ( commandName == "upperlimits" )
            {
               curPair = params->GetDataValuePair("upperlimits");
               curPair->GetData( upperlimits );
            }
            else if ( commandName == "lowerlimits" )
            {
               curPair = params->GetDataValuePair("lowerlimits");
               curPair->GetData( lowerlimits );
            }
            else if (commandName == "totalmatingevents")
            {
               curPair =
                  params->GetDataValuePair("totalmatingevents");
               curPair->GetData( totalmatingevents );
            }
            else if (commandName == "totalruns")
            {
               curPair = params->GetDataValuePair("totalruns");
               curPair->GetData( totalruns );
            }
            else if(commandName == "interactiontype")
            {
               curPair =
                  params->GetDataValuePair("interactiontype");
               curPair->GetData( interactiontype );
            }
            else
            {
               curPair =
                  params->GetDataValuePair("checkingcreature");
               curPair->GetData( checkingcreature );
            }
         }
         std::cout<<"Got XML commands "<<std::endl;
```

Fig. 6.7 Setparams Function of the computational unit

Initial population selection. Before the evolutionary optimization runs starts the values of both low and high-level parameters are supplied to the EA class. Based on the interaction type (manual or interactive), the evolutionary algorithm run number, and the image file the EA population gets loaded. The code fragment that accomplishes this task is shown in Fig. 6.8.

```
Imagefilename = cfdVeParamsManager::getInstance().GetVeParams()-
>GetImageFileName();
std::ostringstream temp_file_name;

if(InteractionType == 1)
{
 temp_file_name<<imagefilename<<"_Intrun"<<run
<<"_"<<popsize<<".txt";
}
else
{
 temp_file_name<<imagefilename<<"_run"<< run <<"_"<<popsize<<".txt";
}
fileName = temp_file_name.str();
file.open( fileName.c_str() );
```

Fig. 6.8 Loading Initial Population for the Optimization Runs

The designer may choose to select the population size as per their requirements. Typically a population size of 32 or 64 is recommended. Each solution represents the original image in the form of image segments. If the interaction type is chosen to be 1, then out of the 32 solution population, the tile centers were chosen at random for 29 solutions and the remaining 3 solutions were completely designed by the designer. Depending on the image features and high color variances the designer can pick, move, and add tile centers in such interesting areas and delete the tile centers from the less interesting areas. Nearly 10% of the population was desired to be interactive in this work. Hence, only 3 interactive solutions per population were chosen. The designer developed segments were generated using an interactive design canvas.

Initial fitness evaluation. Once the initial population is supplied the initial fitness values needs to be evaluated. To evaluate the fitness, the given Voronoi generators should first be used to represent the image as 'N' image segments. To segment the original image the Voronoi generators (or tile centers) are needed. Since weighted Voronoi tessellations are used in this work, the numerical weight corresponding to the tile centers are also needed. The numerical weight *r* for the tile center is the square root of variance within a square region. The size of the square region is proportional to the number of tiles used to represent the image. Equation (6.1) is used to estimate the size of the square region, S. Let N is the number of image segments used to represent the image. As the optimization proceeds, the value of N is incremented to decrease the color variance in each image segment. The constant terms *w* and *h* are respectively the width and the height of the original image. The product of *w* and *h* is the number of pixels (N_p).

$$\text{S} = \text{Integer}\left[\sqrt{\frac{N_p}{N}}\right] \tag{6.1}$$

Based on these segments, the fitness value of a solution can be estimated using equation 6.2. The term *Var* in equation (6.2) indicates the color variance in each image segment. During the estimation of initial fitness if the variance value of a segment is above the designer prescribed threshold then an additional Voronoi Generator is added, i.e., the segment size is increased by one. Now, the fitness values of the entire population have been evaluated.

$$\text{Fitness} = \frac{\sqrt{\sum_{i=1}^{N}(Var)_i}}{N} \tag{6.2}$$

The initial population was represented with 50 segments each, i.e., 50 tile centers with their corresponding numerical weights. Hence, pre-optimized initial image segments are created using weighted Voronoi tessellations. At this stage, the test image is represented by 50 segments (or a maximum of 50 unique colors). This may not be sufficient to reproduce the image efficiently. Evolutionary optimization technique is implemented to minimize the segment variance in each segment. This is done by moving and adding tile centers during evolutionary optimization process.

Optimization process. The evolutionary algorithm begins by selecting a parent at random. The co-parent selected using *roulette selection*. Two-point crossover with probabilistic mutation is used. The parent and the co-parent may have different gene lengths. Hence both crossover points are chosen to be within the length of the smaller mating member. The probabilistic mutation has a 50% chance of increasing the weights of 'n' random generator points by 10, and a 25% chance of increasing the number of Voronoi generators up to 10 with newly defined points and weights. The remaining 25% of the mutation probability decreases the weights of 'n' random generator points by 1.The gene length of the solution represents the number of image segments used to represent the image. As the evolution proceeds, the gene length of all the solutions in the population does not remain the same. Depending on the feature density, the designer user can add and remove a number of image segments from a current solution in the evolving population. During the evolutionary optimization run, at regular intervals, the designer can insert a new solution in place of the solution that has the least fitness value. These solutions can be generated on-the-fly using the interactive segmentation canvas without disturbing the EA runs. This tweaked-up solution is sent to the evolving population to guide the evolutionary process.

If the color variance in each image segment approaches zero, then the optimum value is said to be reached. The designer can relax this condition depending on the features that are essential for the decision making. Evolutionary algorithms are used for wide exploration of the search space. This technique has many parameters to update after evolutionary run. Simple and yet powerful stochastic technique in evolution strategies are currently being explored to speed-up the optimization process. Evolutionary strategies can be used on image segments that are obtained from the evolutionary algorithms. Preliminary work on evolutionary stained glass strategies for evolving image segments have given encouraging results to try this technique for speeding up the evolutionary segmentation process [Ashlock et al., 2006].

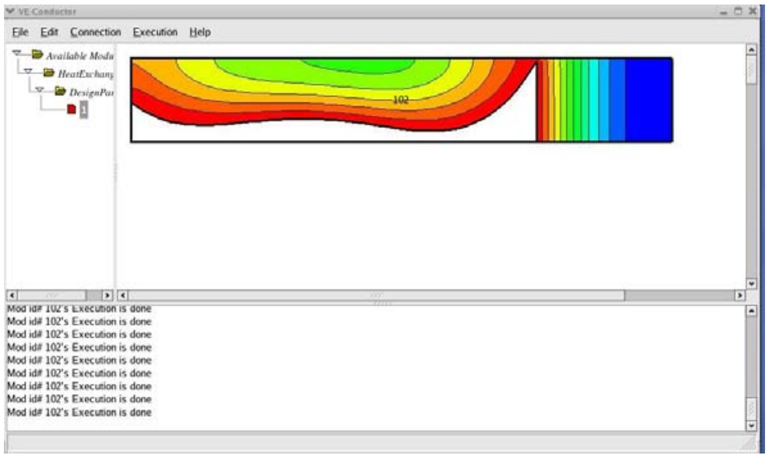
6.3 Interactive Segmentation Visualization

The optimized and segmented image data is stored in a text file. The details available in the text file are the Voronoi generator points, their numerical weights, and their color value. These interim image segment text files are generated at the end of every 10 mating events. If the designer observes if the visual quality of the image segments remaining stagnant as the mating event progress, then suitable preference articulation can be given to the computational unit either directly through the least fit solution replacement or by supplying new initial population. Computational engine (VE-CE) in the VE-Suite controls the segmentation and optimization algorithm. Computational unit is the plugin to the VE-CE. The computational unit uses the interim image segment text files to create the interim image segments. If the designer finds that the interim image segments generated are of sufficient quality for making decisions, then the evolutionary runs can be stopped. The segmented image can be displayed either directly through any image viewing software or in the graphical plugin of the VE_Suite. The quality of the optimized image segments are assessed using the decision quality index of images, developed by [Karthikeyan et al., Submitted]. If the quality of the segmented image segments is not acceptable, then the re-optimization is done by changing the low and high level parameters that form a part of user interface and computational engine.

CHAPTER 7. IMPLEMENTATION OF PROGRESSIVE INTERACTION SYSTEM FOR SHAPE OPTIMIZATION

7.1 Designer Preference Module

Fig. 7.1 shows the main window of the designer preference module for the shape optimization problem. On selecting the icon of the fin the application specific implementation can be accessed. The designer should be aware of the high-level and low-level parameters that need continuous or intermittent interaction. Low level parameters include population size, crossover, mutation, and replacement operators. In addition, the total number of mating events, and the number of evolutionary runs are also provided through the designer preference module. The user interface shown in Fig. 5.7 is used for providing the low-level parameters to the interactive optimization system. The parameters used for this test case must be registered before they can be accessed or modified by the designer. The code fragment used to register the parameters is shown in Fig. 7.2.



```
Mod id# 102's Execution is done
Mod id# 102's Execution is done
Mod id# 102's Execution is done
Mod id# 102's Execution is done
Mod id# 102's Execution is done
Mod id# 102's Execution is done
Mod id# 102's Execution is done
Mod id# 102's Execution is done
Mod id# 102's Execution is done
Mod id# 102's Execution is done
```

Fig. 7.1 Designer Preference Module – Main Window

```
HeatExchanger::HeatExchanger( )
  {
      name = "HeatExchanger";

      // Register the parameters that need to sent to the Unit
      RegistVar("mutationrate", &mutationrate);
      RegistVar("crossoverrate", &crossoverrate);
      RegistVar("popsize", &popsize);
      RegistVar("replacemethod", &replacemethod);
      RegistVar("crossovermethod", &crossovermethod);
      RegistVar("mutationmethod", &mutationmethod);
      RegistVar("checkinggene", &checkinggene);
      RegistVar("terminationmethod", &terminationmethod);
      RegistVar("designparamnames", &designparamnames);
      RegistVar("designparams", &designparams);
      RegistVar("upperlimits", &upperlimits);
      RegistVar("lowerlimits", &lowerlimits);
      RegistVar("activeindexs", &activeindexs);
      RegistVar("inactiveindexs", &inactiveindexs);
      RegistVar("checkingcreature", &checkingcreature);
      RegistVar("totalruns", &totalruns);
      RegistVar("totalmatingevents", &totalmatingevents);
      RegistVar("degree", &degree);
      RegistVar("interactiontype", &interactiontype);
}
```

Fig. 7.2 Registeration of Parameters for Shape Optimization

High level parameters. These are the problem-specific parameters. For instance, in this test case, the high level parameters are the fin spacing, the fin base thickness, fin length (x coordinates), fin degree (n), and the y-coordinates ((n+1) points, along the fin length). Except the constant parameters, in this case, the fin length (0.75), all the other high-level parameters make up the gene for the evolutionary optimization. The user interface begins with the input of fin profile degree and the interaction type. If the fin profile degree is chosen to be 3, then a total of 4 y values exist, i.e., Y_0, Y_1, Y_2, and Y_3. After providing these two values, this module automatically generates the design setup file. The code fragment that automatically generates the design setup file based on the fin degree is shown in Fig. 7.3. This function also uploads the high-level parameters in the right side of the high-level parameter setup dialog on selecting “Load Parameters from File” button, as shown in Fig. 7.4. The constraints file is automatically generated based on the user specified values of fin degree. The user inputs this constraint file to set the numerical limits of the high-level parameters. The base thickness and fin spacing range between [0,1]. The Y coordinate values range from 5% to 80% of fin spacing.

```
      std::ofstream designsetup;
      designsetup.open("DesignSetup.txt", std::ios::out);
      designsetup<<"BASE_THICKNESS"<<std::endl;
      designsetup<<"FIN_SPACING"<<std::endl;
      for(int i=0; i<(degree+1); ++i)
      {
            designsetup<<"Y_"<<i<<std::endl;
      }
      designsetup.close();

      wxFileDialog dialog
                    ( this,
                      _T("Load design setup file"),
                      _T(""),
                      _T(""),
                      _T("All files (*.*)|*.*")
                    );

    dialog.SetDirectory(wxGetHomeDir());

    if (dialog.ShowModal() == wxID_OK)
    {
      wxString info;
      info = dialog.GetPath().c_str();
      std::ifstream designsetupfile;
      designsetupfile.open(info);
      std::string temp_string;
      while(designsetupfile>>temp_string)
      {
         m_lbox->Append(temp_string.c_str());
      }
      designsetupfile.close();
    }
```

Fig. 7.3 Design Setup File Generation

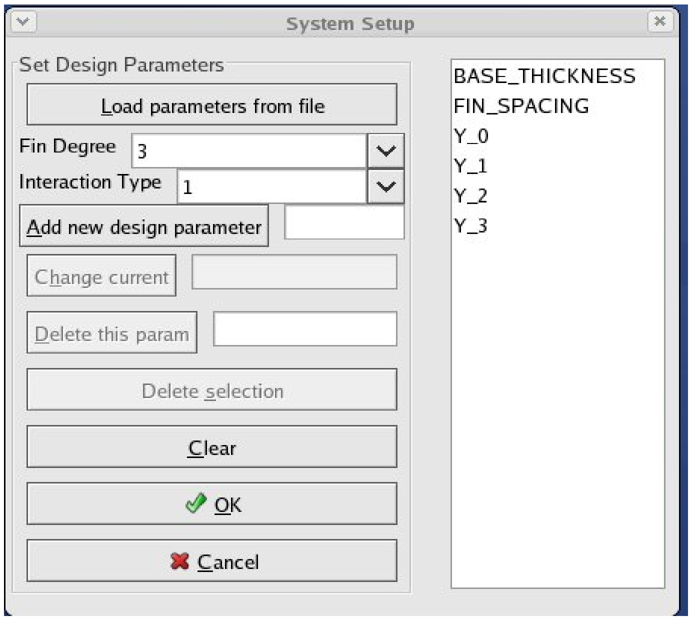


Fig. 7.4 High-level Parameter Setup Dialog

If the interaction type is chosen to be 1 then the designer has the flexibility of generating his/her own population based on his specific knowledge. This gives a method of isolating the parameters that prior knowledge or interactive design have shown do not significantly contribute to the system performance. If the interaction type is chosen to be 0 then the designer accepts the random initial population.

Interactive fin profile design canvas. Fig. 5.8 displays the interactive profile selection canvas developed as a part of this research. This interactive canvas was developed using wxWidgets and OpenGL. The designer needs to provide the number of fin profiles that need to be generated using this interactive canvas and the degree of each fin profile. Based on the profile degree the Y values get enabled or disabled accordingly. All the high-level parameters, i.e., Y values, fin spacing and fin base thickness can be interactively selected using sliders. Once after all the required selections are done, the fin profile is drawn in the design canvas window. This gives the designer an idea if the fin profile that they are interested may or may not work. In Fig. 5.8, the fin profile is shown in solid line and the dotted black lines show the constraints. The designer can write the high-level parameters into a file that makes up the gene for evolutionary optimization runs. Multiple fin profiles can be added on this canvas by clearing the screen and providing the new set of high-level parameters. This information can be written out as a separate file or as an appendage to the existing interactive fin profile file.

7.2 Interactive Optimization Module

The variables registered in Fig. 7.2 are sent to the computational unit. The code fragment similar to the one shown in Fig. 5.10 accepts the variables. As presented in the Interactive Optimization Module of Chapter 5, the evolutionary optimization runs need both low-level and high-level parameters. Based on the interaction type: random or interaction the initial population files get loaded. The population files are decided based on the fin degree, and the run number. The fitness values of the fin profiles are estimated in terms of a dimensionless Nusselt number as described in Chapter 4. The fitness values are normalized in the range of [0,1]. The fitness values are determined using the numerical solver originally developed by Suram et al., [2006]. The call to the numerical solver from the optimization module is shown in Fig. 7.5.

```
double cfdCalcCaseImpForModel::FindFitness(std::vector<double>
values)
{
      double temp;
      double *Yvalues;
      int size = values.size();
      int degree = (size – 3);
      Yvalues = new double[degree+1 ];

   for(int i=0; i<(size-2); ++i)
   {
      Yvalues[i] = values[i+2];
   }

   RunModel run( (size-3),0.75,values[0],values[1],Yvalues );

   NewVectorValues.clear();
   NewVectorValues = run.GetNewParamValues( );
   temp = run.GetFitness( );

   std::cout<<"Fitness from FindFitness is :"<<temp<<std::endl;
   delete [] Yvalues;
   return temp;

}
```

Fig. 7.5 Call for Fitness Evaluation

The class RunModel accepts the fin profile values as per the constraints. This class divides the fin model shown in Fig. 4.5 into 6 parts, three parts along the fluid region, one part along the fin length, and two parts along the fin base. The RunModel class acts as the master class to generate grid, solve the fluid flow, and energy equations to determine the fitness values. The fitness values are normalized in the range of [0, 1]. The designer specified profiles are free from abrupt changes in curvature. Hence, the fitness values are within the normalized. However, for the fin profiles generated using random population had the abrupt change in curvature resulting in the fitness value equal to “Nan”. To address this issue, the problematic profiles are revisited within the RunModel class using new profile values. The RunModel class also develops the data file and the VTK file for each fin profile during initial fitness evaluation and during evolutionary process. During the optimization process, the designer has the option to “inject” their knowledge by replacing a low performing member in the population with their choice of solution. This is done at specified locations, say after every 10 mating events.

7.3 Interactive Visual Analysis

The VTK files are generated in the computational unit after every mating event. The best VTK files after every 10 mating events are supplied to the graphical engine where the results are shown in an immersive virtual environment. The VTK files will be loaded as per designers’ input from the GUI. The progress of the evolutionary algorithms would influence the designer’s reaction. If the fitness progress report in the interactive environment shows no improvement, then the designer is prompted to develop a better gene structure for a few solutions in the population and retry the evolutionary process. If the fitness progress is contributed by some solutions, then the designer can select and copy the solutions that are responsible for tremendous fitness growth. A sample visual output of a third degree fin profile with temperature distribution is shown in the Fig. 7.6. The red regions in the Fig. indicate maximum temperature and the blue regions indicating the minimum temperature.

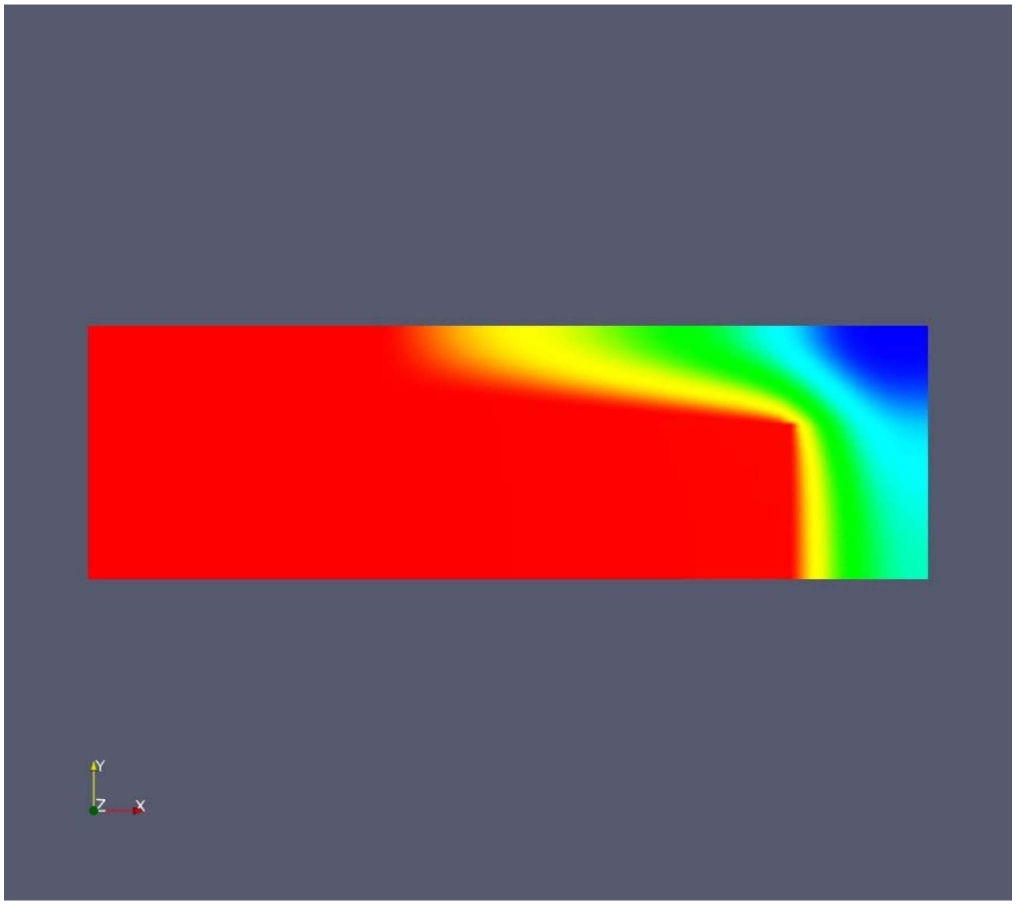


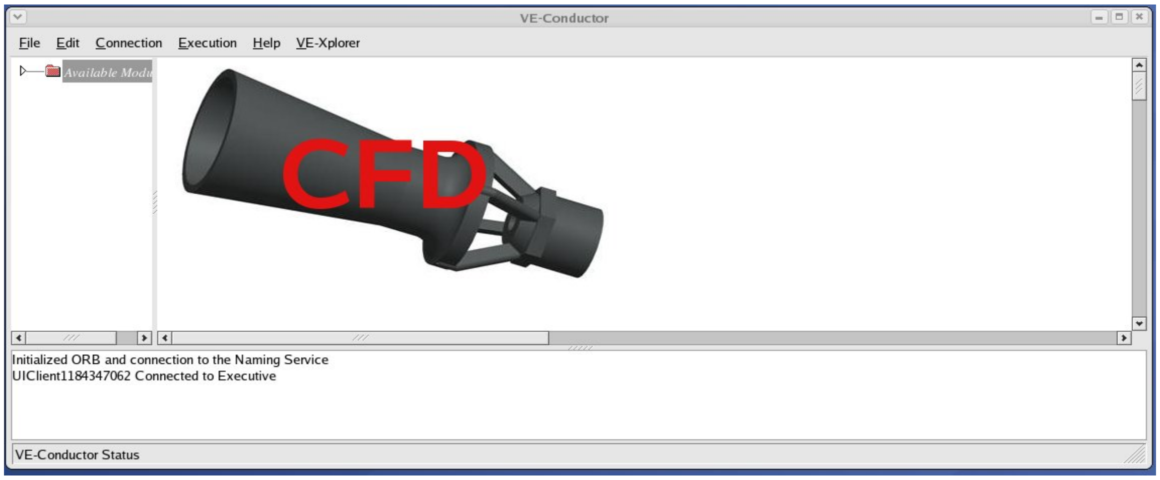
Fig. 7.6 Temperature Distribution in a Third Degree Fin Profile

CHAPTER 8. IMPLEMENTATION OF PROGRESS INTERACTION SYSTEM FOR NOZZLE OPTIMIZATION

As discussed for the other two test cases, the designer preference module and computational unit are implemented for the nozzle problem. The designer preference module in this test case is implemented in such a way that the nozzle evolutionary optimization runs can start from any intermediate locations, in terms of solutions initial fitness evaluation, mating events, and evolutionary runs. The new feature added to the computational unit is that in addition to accepting to start the runs from an intermediate location, this unit also accepts the designers' choice of parents for evolutionary mating events. The details regarding the implementation are presented in the rest of this Chapter.

8.1 Designer Preference Module

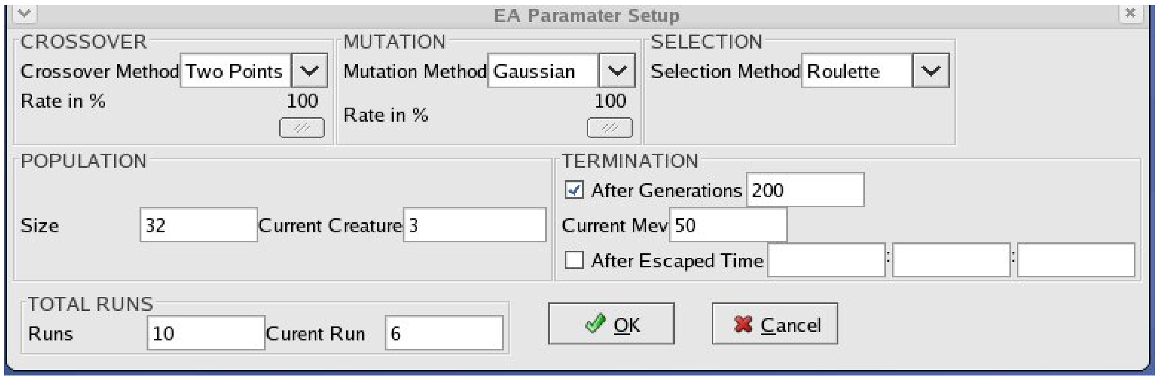
Fig. 8.1 shows the launching window for the designer preference module for this problem. On selecting on the nozzle icon in this window all the application specific parameters can be accessed by the designer. The first step in the implementation is to register the variables used in this work, similar to the one shown in Fig. 7.2. The second step is to identify the low level and high level parameters for the optimization. The low-level parameters discussed for the other two test cases are applicable for this case. In addition, the details regarding the current run, current mating event number, and current creature are needed. This is presented in Fig. 8.2.



```
Initialized ORB and connection to the Naming Service
UIClient1184347062 Connected to Executive
```

VE-Conductor Status

Fig. 8.1 Designer Preference Launcher for Nozzle



CROSSOVER
Crossover Method: Two Points
Rate in %: 100

MUTATION
Mutation Method: Gaussian
Rate in %: 100

SELECTION
Selection Method: Roulette

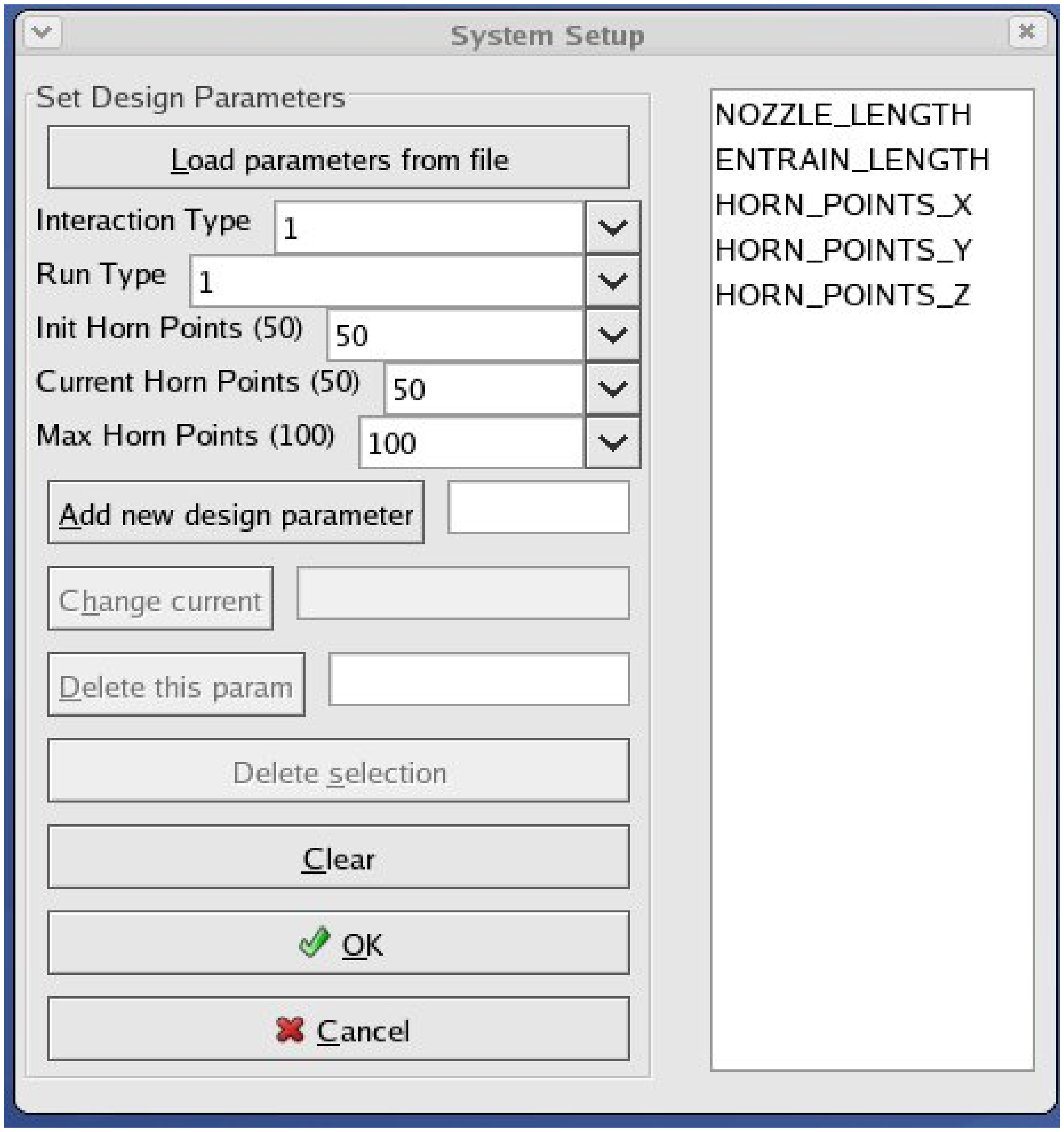
POPULATION
Size: 32
Current Creature: 3

TERMINATION
[✓] After Generations: 200
Current Mev: 50
[ ] After Escaped Time: : :

TOTAL RUNS
Runs: 10
Curent Run: 6

Fig. 8.2 Low-level Parameter Setup Dialog

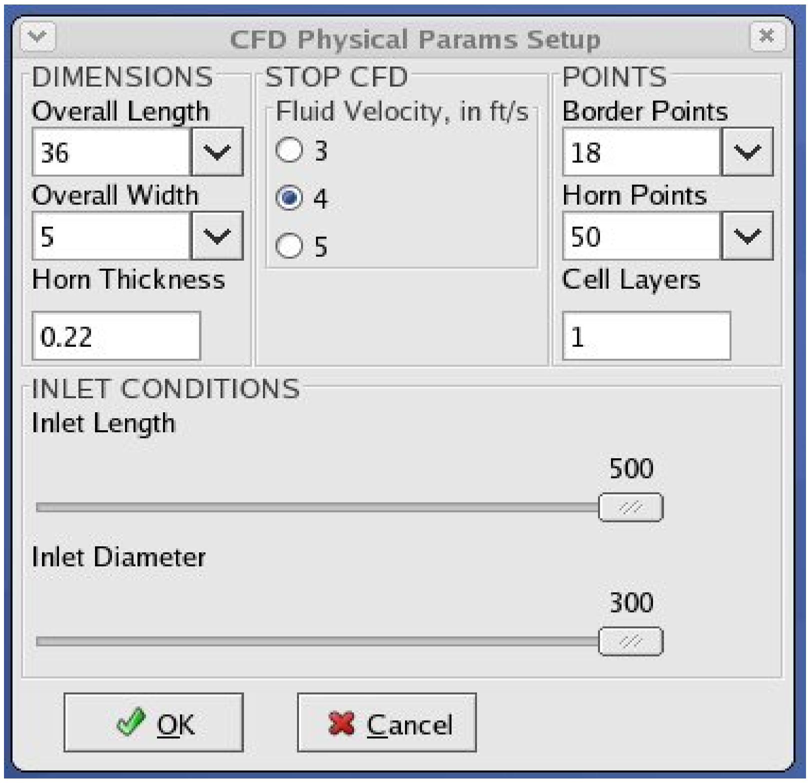
As shown in Fig. 8.2, using this information, the computational unit can start the run number 6 starting with the mating event number 50. However, the high level parameters for this case include interaction type, current and initial horn points, maximum horn points, nozzle length, entrainment length, and the horn point data. The details of the high level parameters are presented in Fig. 8.3. Unlike the high level parameter user interface of the other two cases, this case asks the designer to specify the “RunType”. If this type is chosen to be 1, then the evolutionary optimization can start from any specified location that the user wants. In addition, the designer can choose a variety of physical parameters, mesh parameters, and flow parameters as shown in Figs. 8.4-8.7.



Interaction Type: 1
Run Type: 1
Init Horn Points (50): 50
Current Horn Points (50): 50
Max Horn Points (100): 100

NOZZLE_LENGTH
ENTRAIN_LENGTH
HORN_POINTS_X
HORN_POINTS_Y
HORN_POINTS_Z

8.3 Designer Interface for High-level Nozzle Parameters


CFD Physical Params Setup

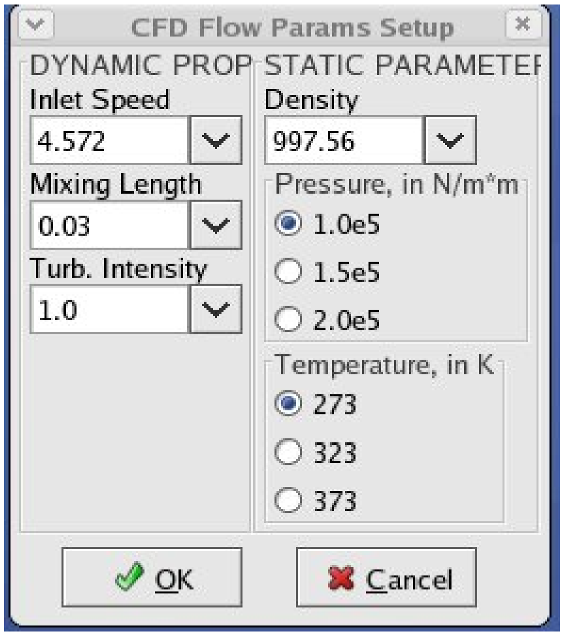
DIMENSIONS
Overall Length: 36
Overall Width: 5
Horn Thickness: 0.22

STOP CFD
Fluid Velocity, in ft/s: ○ 3 [✓ ● 4] ○ 5

POINTS
Border Points: 18
Horn Points: 50
Cell Layers: 1

INLET CONDITIONS
Inlet Length: 500
Inlet Diameter: 300

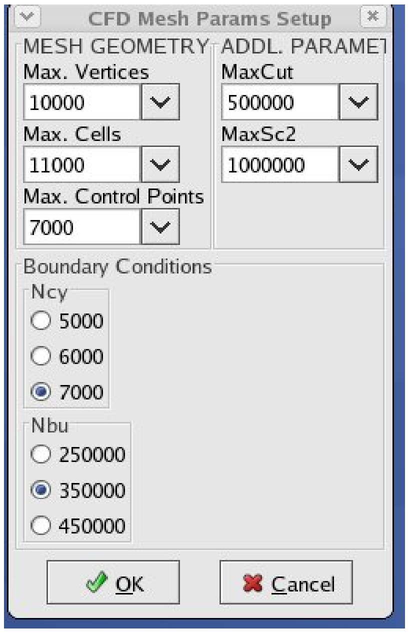
Fig. 8.4 CFD Physical Parameters Setup Dialog


CFD Flow Params Setup

DYNAMIC PROP
Inlet Speed: 4.572
Mixing Length: 0.03
Turb. Intensity: 1.0

STATIC PARAMETEF
Density: 997.56
Pressure, in N/m*m: [✓ ● 1.0e5] ○ 1.5e5 ○ 2.0e5
Temperature, in K: [✓ ● 273] ○ 323 ○ 373

Fig. 8.5 CFD Flow Parameters Setup Dialog


CFD Mesh Params Setup

MESH GEOMETRY

Max. Vertices: 10000

Max. Cells: 11000

Max. Control Points: 7000

ADDL. PARAMET

MaxCut: 500000

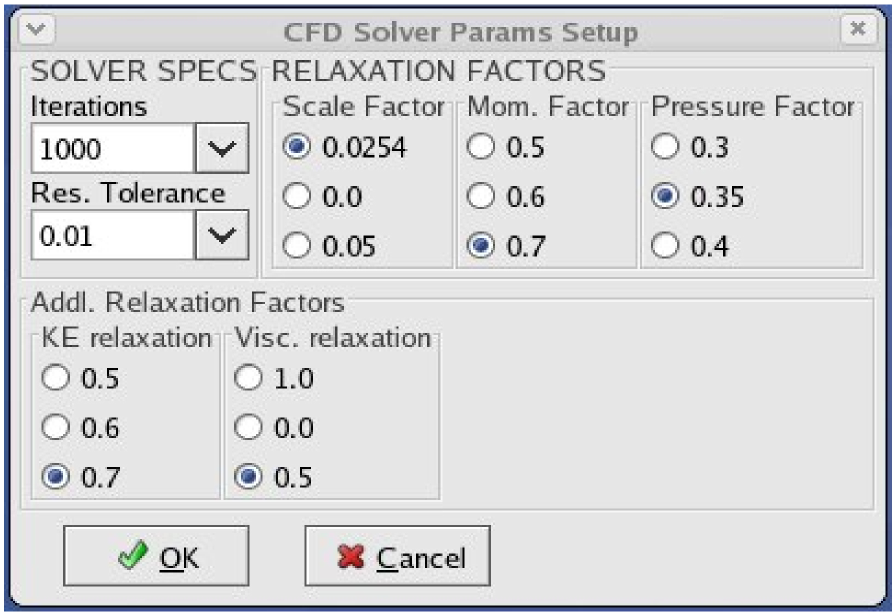
MaxSc2: 1000000

Boundary Conditions

Ncy: ○ 5000 ○ 6000 [✓ 7000]

Nbu: ○ 250000 [✓ 350000] ○ 450000

Fig. 8.6 CFD Mesh Parameters Setup Dialog


CFD Solver Params Setup

SOLVER SPECS

Iterations: 1000

Res. Tolerance: 0.01

RELAXATION FACTORS

Scale Factor: [✓ 0.0254] ○ 0.0 ○ 0.05

Mom. Factor: ○ 0.5 ○ 0.6 [✓ 0.7]

Pressure Factor: ○ 0.3 [✓ 0.35] ○ 0.4

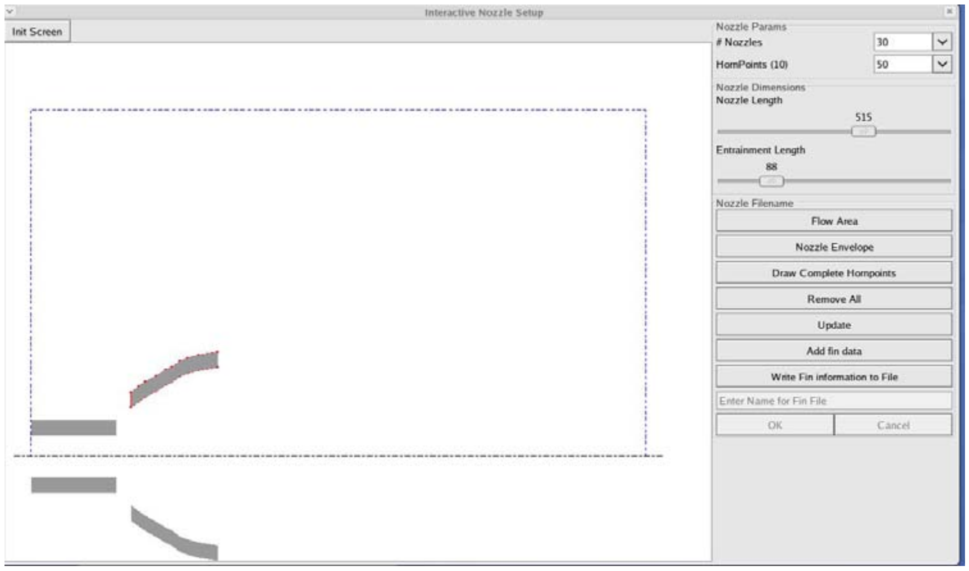
Addl. Relaxation Factors

KE relaxation: ○ 0.5 ○ 0.6 [✓ 0.7]

Visc. relaxation: ○ 1.0 ○ 0.0 [✓ 0.5]

Fig. 8.7 Solver Parameters Setup Dialog

Interactive nozzle profile canvas. Fig. 8.8 displays the interactive nozzle profile canvas developed as a part of progressive interaction. This canvas is enabled before, during, and even after the evolutionary optimization runs are completed. The designer needs to specify the number of horn points, nozzle length and entrainment length. The horn thickness, the parameters like inlet length, and inlet diameter are provided in the CFD physical parameters are accessed by this canvas.



Nozzles: 30
HornPoints (10): 50
Nozzle Length: 515
Entrainment Length: 88

Fig. 8.8 Interactive Nozzle Profile Canvas

The canvas shows the flow envelope, which is 36”x5”, within this flow envelope the nozzle is placed. Another envelope (for nozzle horn profile) was displayed initially. This envelope gave the designer the information on where exactly their horn points have to be located. Based on this, the designer can mouse click on the locations to generate the nozzle horn profile as shown in the Fig. 8.8. The canvas is also equipped with the file handling routines to generate population files consisting of these interactive nozzles generated by the designer.

8.2 Interactive Optimization Module

As shown in Chapters 6 and 7, the variables are registered before they are sent to the computational unit. As presented in the Interactive Optimization Module of Chapter 5, the evolutionary optimization runs need both low-level and high-level parameters. The designer input of the high-level parameters for the nozzle optimization is done as presented in Figs. 8.3 to 8.9. Based on the interaction type and the number of nozzle horn points the corresponding initial population files get loaded. The fitness values of the nozzle profiles, the magnification ratios are estimated using equation (4.8). This interactive optimization module is integrated with the "nozzle" solver class developed by Engelbrecht [To be submitted]. The nozzle solver defines the boundary points for generating the mesh around the flow area. Prior to defining the boundary points for generating mesh the nozzle profile horn points have to be defined. For the random nozzle population the horn points are generated as a part of the nozzle solver. The design parameters: nozzle length, entrainment length and the coordinate values of the horn points are automatically assigned. In contrast, for the interactive nozzle population the horn points are defined based on user input using the design canvas as presented in Fig. 8.8. By developing a simple function the capability of the nozzle solver is extended to handle designer interactive parameter values. The code fragment presented in Fig. 8.9 illustrates the implementation of "NozzleParamsForFitnessEstimation" function.

```
void Nozzle::NozzleParamsForFitnessEstimation
(std::vector<double> values, int id)
{
   int i, j, numparams;

   numHornPoints = values.size();
   numHornPoints = (numHornPoints-2)/3;
   numparams  =  cfdVeParamsManager::getInstance().GetVeParams()-
>GetDesignParamsNum();

   j = 2;
   nozzleLen = values.at(0);
   entrainLen = values.at(1);

   for(i=0; i<numHornPoints; ++i)
   {
      while( j < ( (3*i) + numparams) )
      {
         if( (j-2)%3 == 0)
         {
            hornPoints[i+1]->SetX( values.at(j) );
         }
         else if( (j-2)%3 == 1)
         {
            hornPoints[i+1]->SetY( values.at(j) );
         }
         else if( (j-2)%3 == 2)
         {
            hornPoints[i+1]->SetZ( values.at(j) );
         }
         j++;
         std::cout<<std::endl;
      }
   }

}
```

Fig. 8.9 Implementation to Handle Designer Interactive Nozzle Population

The driver function that communicates with the nozzle solver class is developed as a part of the interactive optimization module. The code fragment shown in Fig. 8.10 contacts the nozzle solver and all its methods using the nozzle class object (NozzList). It is worth noting that the “NozzleParamsForFitnessEstimation” code presented in Fig. 8.9 is called from the code shown in Fig. 8.10.

```
double cfdCalcCaseImpForModel::FindFitness(std::vector<double>
values, int id)
{
   double temp;
   int run, currmev, currhornpoints;
   std::ofstream tempfile;

   currmev = cfdVeParamsManager::getInstance().GetVeParams()-
>GetCurrentMatingEvent( );
   run = cfdVeParamsManager::getInstance().GetVeParams()-
>GetCurrentRun( );
   hp = cfdVeParamsManager::getInstance().GetVeParams()-
>GetCurrentHornPoints();

   if(currmev == -1)
   {
      MakeNozzleDirectory(id);
   }

   tempfile.open( "setup", ios::out);
   tempfile << "#!/bin/csh -f" << endl;
   tempfile << "cd ./nozzle" << id << "/run_" <<run<<"/"<< endl;
   tempfile.close();

   tempfile.open( "run", ios::out);
   tempfile << "#!/bin/csh -f" << endl;
   tempfile << "cd ./nozzle" << id << "/run_" <<run<<"/"<< endl;
   tempfile << "source /usr/local/starcd/etc/setstar" << endl;
   tempfile << "$STARDIR/bin/star" << endl;
   tempfile.close();

   NozzList[id]->NozzleParamsForFitnessEstimation(values, id);
   std::cout << "Params supplied to Nozzle class"<<std::endl;

   buildNozzle( id, currmev, run );

   temp = NozzList[id]->finalMag;
   std::cout<<"Fitness from FindFitness is :"<<temp<<std::endl;
   return temp;
}
```

Fig. 8.10 Calls for Nozzle Fitness Evaluation

The code presented in Fig. 8.10 makes individual nozzle directories. In addition, a setup file and run file are also generated based on the nozzle number. The setup file consists of the information required to start the commercial CFD software (STAR-CD). The information that can be found in the setup file include: nozzle number, commands to open the STAR-CD, maximum number of vertices, mesh details, boundary conditions, etc. The buildNozzle( ) function shown in Fig. 8.10 calls the nozzle solver class to setup the border points, write nozzle mesh, write boundary conditions, save the .ccm model, execute and run the STAR-CD. A typical setup file for the nozzle is shown in Appendix. The run file sources the installation directory to start the STAR-CD solver.

The evolutionary mating operations should take place to evolve an optimum nozzle configuration. In this work, the roulette wheel selection is used to select the parent and the co-parent. The roulette replacement is used to select the child1 and child2. In addition to these roulette wheel selection, the designer can also free choose the parents for the evolutionary mating operation. The Fig. 8.11 shows the MatingOperation function that clearly illustrates the forward and reverse crossover to produce new nozzle configurations that go in as child1 and child2 respectively. Thus a two point crossover was used for this work. The best nozzle profile after each evolutionary run is stored as a GIF file. These GIF files can be viewed externally or can be visualized through the graphical plugin.

```
void cfdCalcCaseImpForModel::MatingOperations(int parent1, int 
parent2, int child1, int child2)
{  Nozzle *MatingNozzle;
   float crossLen;
   int currmev = cfdVeParamsManager::getInstance().GetVeParams()-
>GetCurrentMatingEvent( );
   int run = cfdVeParamsManager::getInstance().GetVeParams()-
>GetCurrentRun( );

   MatingNozzle = new Nozzle(hp, ch);
   crossLen = MatingNozzle->chooseCrossOver(NozzList[parent1], 
NozzList[parent2]);
   // Forward Crossover For Child 1
   if( crossLen != -1.0f){
      //NozzList[child1]->removeOldFiles( child1, run );
      NozzList[child1]->crossOver(NozzList[parent1], 
NozzList[parent2], child1, crossLen);
      std::cout << "Complete forward crossover function for MEN " 
<< currmev << endl;
   }
   else{
      NozzList[child1]->removeOldFiles( child1, run  );
      NozzList[child1]->defineHornPoints(hp);
   }
   buildNozzle(child1, currmev, run);
   if ( NozzList[ child1 ]->convergeFlag < 0 )
   {  mutateNozzle( child1, currmev, run ); }
   else
   {  LoadCreatures(child1);  }
   matingHist << "Recomputed child (forward cross)" << child1 << " 
Mag: " << NozzList[ child1 ]->finalMag << endl;
   // Reverse Crossover For Child2
   if( crossLen != -1.0f)
   {
      //NozzList[child2]->removeOldFiles( child2, run );
      NozzList[child2]->crossOver(NozzList[parent2], 
NozzList[parent1], child2, crossLen);
      std::cout << "Complete forward crossover function for MEN " 
<< currmev << endl;
   }
   else{
      NozzList[child2]->removeOldFiles( child2, run  );
      NozzList[child2]->defineHornPoints(hp);
   }
   buildNozzle(child2, currmev, run);
   if ( NozzList[ child2 ]->convergeFlag < 0 ){
      mutateNozzle( child2, currmev, run );
   }
   else {
      LoadCreatures(child2);
   }
}
```

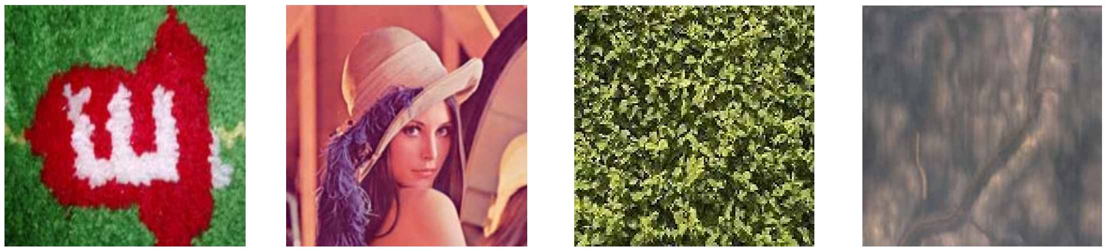
Fig. 8.11 Code Fragment for Evolutionary Mating Operations

CHAPTER 9. RESULTS AND DISCUSSION

In this research work progressive designer interaction has been implemented with two prime goals, they are, to determine the best possible engineering solution under the given problem setup and constraints, and to expedite the engineering optimization process. With these goals in place, this chapter is dedicated to study the results available on the test cases that were used to demonstrate the progressive interaction. Parts (a) through (c) are the three dedicated sessions that present and discuss the results available from all the three test cases.

Part (a) – Image Segmentation and Optimization

The results presented in this section are centered on two major areas. 1. The progress of evolutionary algorithms in terms of fitness values and 2. The visual quality of the image segments during and after the evolutionary runs. Since the image segment optimization is a minimization problem, the lower fitness values are desired. The efficacy of the progressive designer interaction on image segmentation and optimization is demonstrated by using four test images. The test images are shown in the Fig. 9.1. The test images were chosen so that a broad range of image features are covered. The first image is the one with simple features (garden --image id. 1), second image is the most popular test image (Lena -- image id. 2), the third one with most complicated features (greens -- image id. 3), and the fourth one is a complex continuous tone aerial survey image (aerial survey -- image id. 4). The typical binary sizes of these survey images are in the order of three to five megabytes. This image segmentation algorithm was originally proposed for the efficient transfer of aerial survey images to remote and low-bandwidth areas. In addition to satisfactory segmentation of aerial survey images a variety of photographic images were also segmented to demonstrate the wide applicability of this segmentation algorithm. All the test images used for this work were 24 bpp true color images. Each of the RGB color channels had an 8-bit representation. The file size details of the test images in the ASCII format, and pixel sizes are presented in Table 9.1.



(a) Garden (b) Lena (c) Greens (d) Deere

Fig. 9.1 Test Images for Segmentation and Optimization

Table 9.1 Details of the Test Images

Image Id.	Pixel Size	Original file size, (ASC), KB	Allowable Max. Image segments
Garden	162 x 216	417	3000
Lena	512 x 512	3124	8000
Greens	320 x 240	915	5000
Deere	1187x844	11936	10000

As presented in Chapter 4, weighted Voronoi tessellations were used to segment the images according to Euclidean distance of the pixel from the Voronoi generators (or tile centers). The image segments are then optimized using evolutionary algorithms to minimize the color variance such that each segment (a group of pixels) can be represented by one color. A 32 member (or solution) population evolutionary algorithms were used for this study. Initially, each solution represented the test image using 50 image segments. A total of 10 evolutionary runs were conducted using random population, and 10 evolutionary runs were conducted using designer interactive population. In the random population, all the 32 solutions in the population were generated by the computer. In the interactive population, nearly 10% of the solutions were created by the designer using the interactive image segmentation canvas as shown in Chapter 6. In order to study the user-friendliness and effectiveness of the implementation of progressive interaction scheme human subjects' experiments were conducted. A total of 4 human subjects were used, 2 senior level graduate students and 2 entry level graduate students with appreciable knowledge on image processing. So a total of 8 evolutionary runs (2 runs per subject per image) were conducted on each test image. The human subjects provided both the initial interactive population and interactive solution replacement (during the evolutionary runs) using interactive segmentation canvas. For all the evolutionary runs, 200 mating events were allowed. The other stopping criteria includes the maximum number of image segments based on the image pixel size and the average color variance for image segments should be less than or equal to 3.0 (i.e., unit color value variation in the R, G, B channels). The values of maximum allowable image segments for the test images are presented in Table 9.1

In case of human subjects' interaction and designer interaction, the user had an opportunity to replace the existing solution in the population with their choice of solution once in every 10 mating events. The progress of the evolutionary algorithms in terms of fitness values are compared for random, designer interaction and human subjects' interaction evolutionary runs. The results of this comparison are presented in Figs. 9.2-9.5.

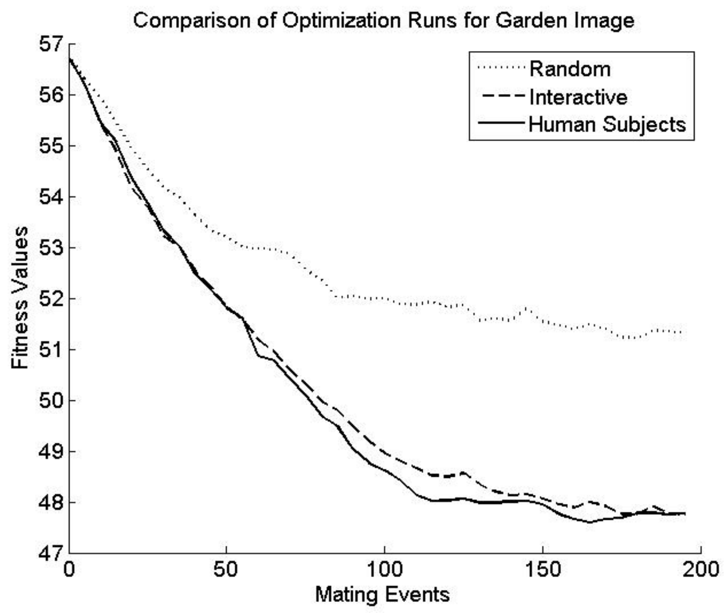


Fig. 9.2 Comparison of Evolutionary Optimization runs for Garden Image

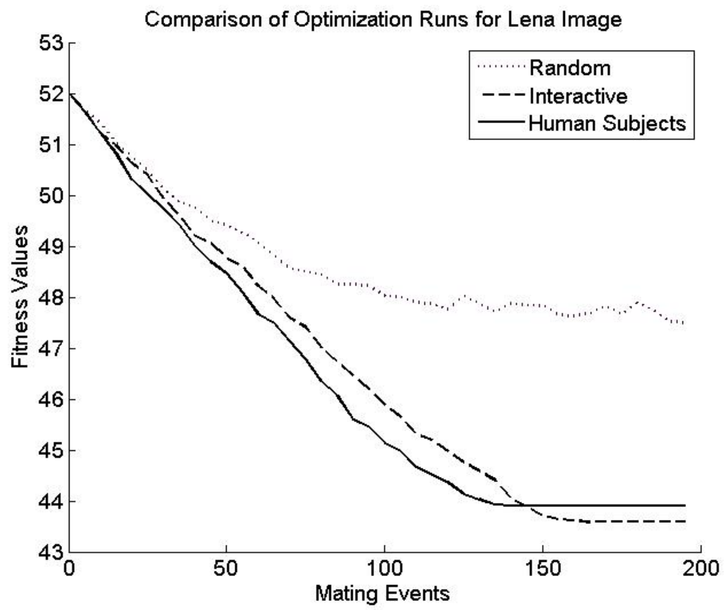


Fig. 9.3 Comparison of Optimization Runs for Lena Image

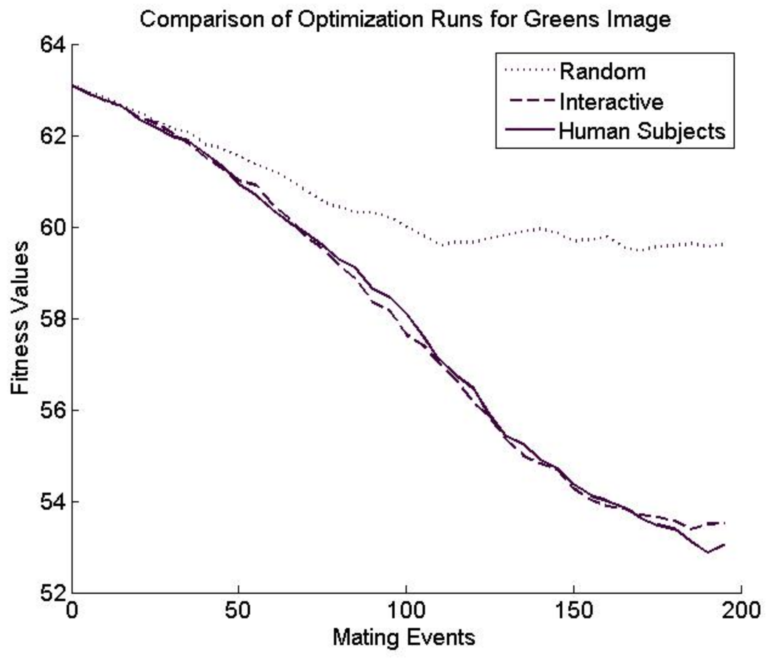


Fig. 9.4 Comparison of Optimization Runs for Greens Image

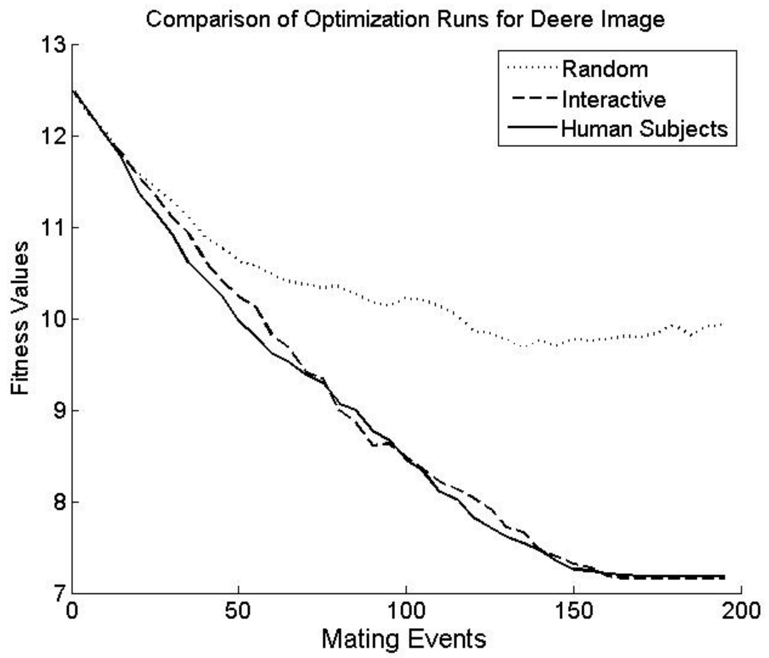
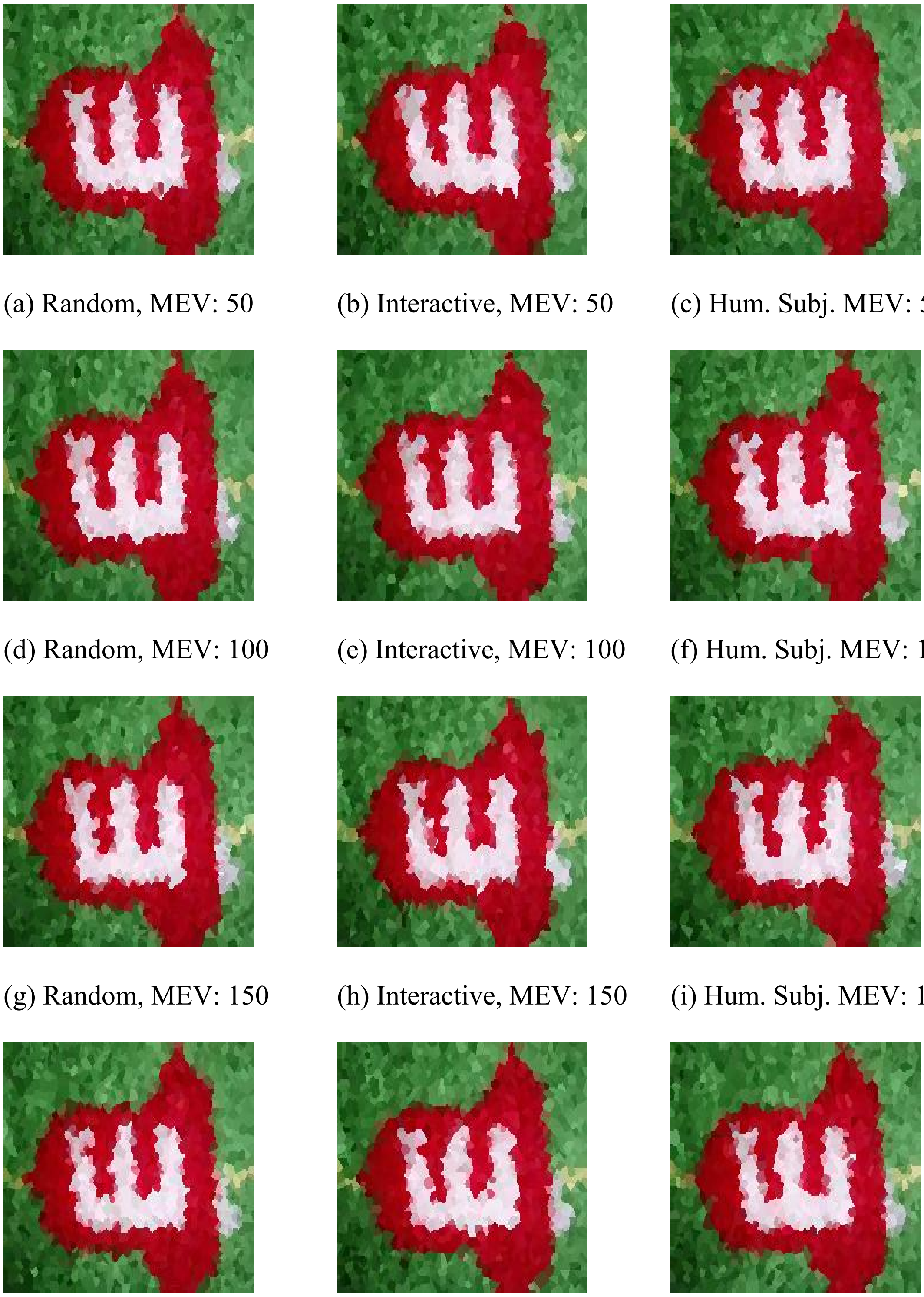


Fig. 9.5 Comparison of Optimization Runs for Deere Image

From Figs. 9.2-9.5, it is evident that the designer interactive and the human subjects' interactive evolutionary runs are clearly ahead of the random runs. Thus the efficacy of the progressive designer interaction in terms of progress of fitness values are demonstrated in these Figs. It is understood that the average initial fitness of all the random, designer interaction and human subjects' interaction runs are comparable. As the evolutionary algorithm moved forward the designer and human subjects were "smart" to learn the search space and were able to capture the regions of the solution space that reduced the image segment color variance. This was primarily achieved by adding more Voronoi generators in the region where the designer perceived of high feature density. For all the four test images, the performance of the designer interaction and the human subjects' interaction were almost comparable. This indicates that the implemented progressive interaction scheme gives consistent results when the user clearly understands the underlying optimization concepts.

The user was "smart" to move the Voronoi generators to the region of high details and add more number of Voronoi generators whenever required. Adding more generators reduced size of the image segments and hence the segment color variation. This resulted in higher fitness values of the interactive runs. The appreciable image quality is required for the designer to make use of these images and make decisions. During the evolutionary runs the user needs to ensure that the optimization produces image segments with acceptable quality so that the decisions can be made on-the-fly. To confirm this, the interim image segments (after 50, 100, and 150 mating events) are compared. The results of this comparison are shown in Fig. 9.6-9.9.



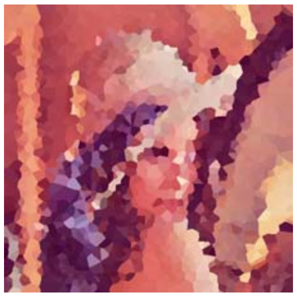
(a) Random, MEV: 50 (b) Interactive, MEV: 50 (c) Hum. Subj. MEV: 50

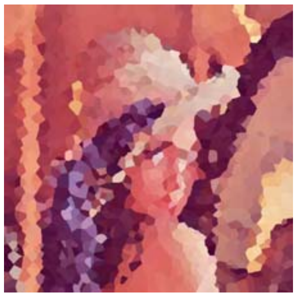
(d) Random, MEV: 100 (e) Interactive, MEV: 100 (f) Hum. Subj. MEV: 100

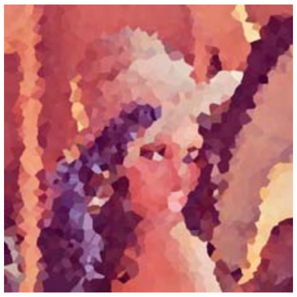
(g) Random, MEV: 150 (h) Interactive, MEV: 150 (i) Hum. Subj. MEV: 150

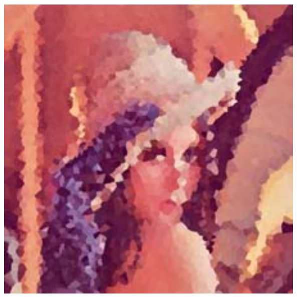
(j) Random (final) (k) Interactive (final) (l) Hum. Subj. (final)

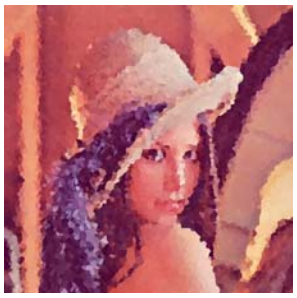
Fig. 9.6 Comparison of Garden Image Segments During Evolutionary Optimization

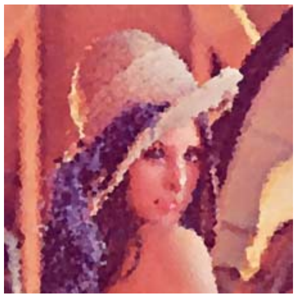

(a) Random, MEV: 50

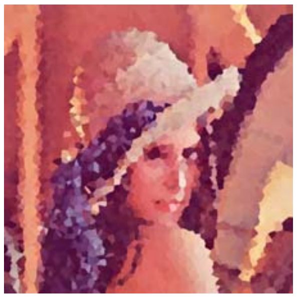

(b) Interactive, MEV: 50

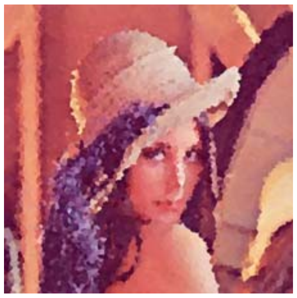

(c) Hum. Subj. MEV: 50

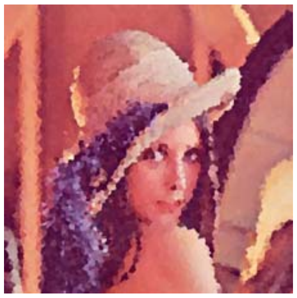

(d) Random, MEV: 100

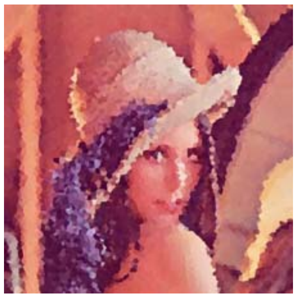

(e) Interactive, MEV: 100

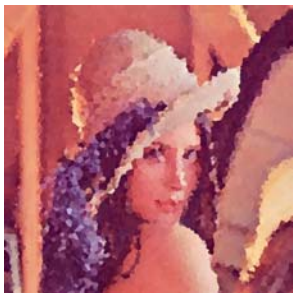

(f) Hum. Subj. MEV: 100


(g) Random, MEV: 150


(h) Interactive, MEV: 150


(i) Hum. Subj. MEV: 150


(j) Random (final)


(k) Interactive (final)

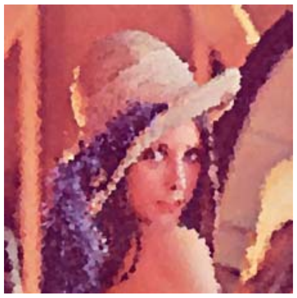
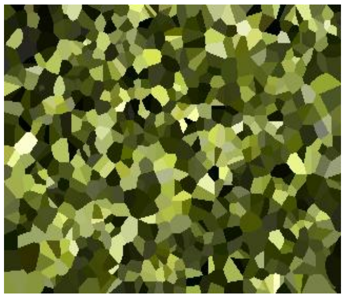
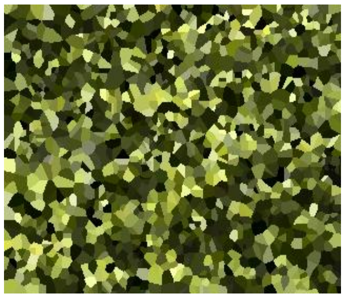
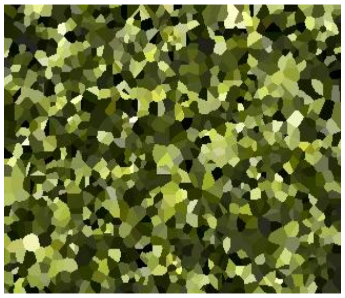
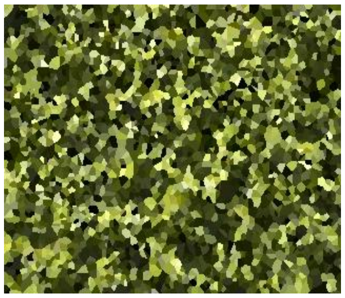

(l) Hum. Subj. (final)

Fig. 9.7 Comparison of Lena Image Segments During Evolutionary Optimization

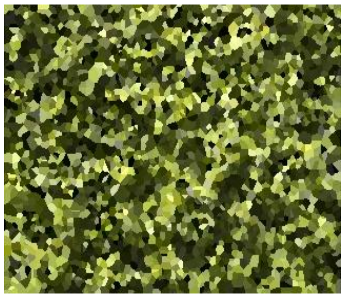

(a) Random, MEV: 50

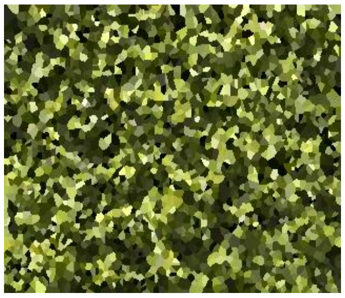

(b) Interactive, MEV: 50

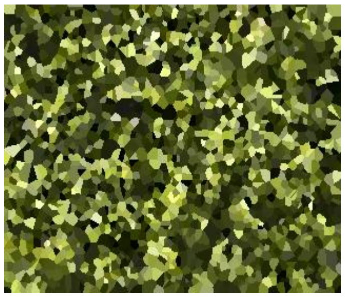

(c) Hum. Subj. MEV: 50

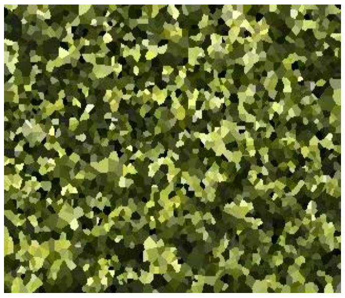

(d) Random, MEV: 100

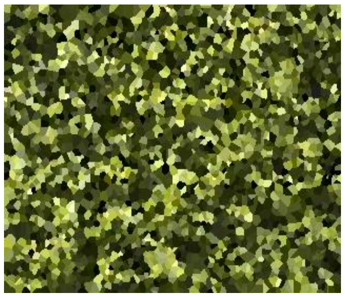
(e) Interactive, MEV: 100

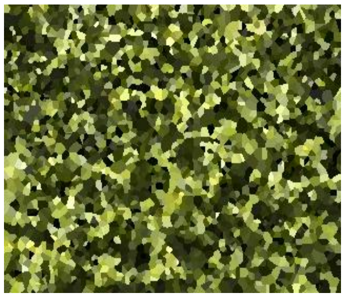

(f) Hum. Subj. MEV: 100


(g) Random, MEV: 150


(h) Interactive, MEV: 150


(i) Hum. Subj. MEV: 150


(j) Random (final)


(k) Interactive (final)

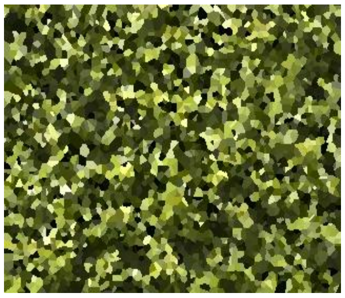

(l) Hum. Subj. (final)

Fig. 9.8 Comparison of Greens Image Segments During Evolutionary Optimization

(a) Random, MEV: 50

(b) Interactive, MEV: 50

(c) Hum. Subj. MEV: 50

(d) Random, MEV: 100

(e) Interactive, MEV: 100

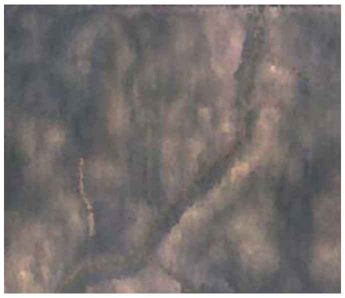
(f) Hum. Subj. MEV: 100

(g) Random, MEV: 150

(h) Interactive, MEV: 150

(i) Hum. Subj. MEV: 150

(j) Random (final)


(k) Interactive (final)

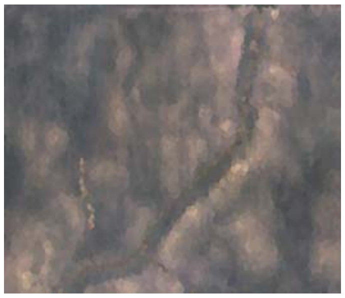

(l) Hum. Subj. (final)

Fig. 9.9 Comparison of Aerial Image Segments During Evolutionary Optimization

From the results available from Figs. 9.6-9.9 it is understood that the user interaction plays a significant role as the image size increases. For example, in 216x162 garden image though the fitness values of the interactive images (both designer and human subjects) are higher than that of the random image, very appreciable image quality variation can be felt. However, in big images like Lena and aerial survey, very stark difference in the visual image quality can be felt as early as 50 mating events. And as the mating events increased the visual image quality from the random runs tend to catch up with that of the interactive runs. Despite this the visual quality of the interactive images are better than the image segments generated by random runs. Thus the results obtained from Figs. 9.6-9.9 are consistent with that from Figs. 9.2-9.5.

Conclusion and Recommendations for Future Work

This part presented the results of the image segment optimization using the progressive designer interaction. Image segmentation is a non-trivial problem and multiple feasible solutions do exist. The results available so far establish that the designer interaction guides the evolutionary algorithms to move towards the higher fitness regions during the early stages of evolutionary process. As discussed earlier, the results of the EA depend on the quality of the initial solutions and the path the EA traverses to reach the solution. Since this problem is a variable length EA problem, the designer interaction mostly increased the number of image segments thus resulting in more time to complete the mating events in case of the interactive runs when compared to that of the random runs.

The future work or enhancement research in this test case include: Use of curve fitting algorithms within each image segment to reduce the evolution time especially for the interactive segment optimization runs. In addition, evolutionary stained glass clustering strategies are being studied as an enhancement to the evolutionary algorithm based segmentation. Currently, the optimization module generates the output image segments for every 10 mating events. They are viewed externally by using a variety of commercial and non-commercial software. These image segment outputs can be displayed in the graphical plugin of the VE-Suite. The control to load these image segments would come from the designer preference module. Decision quality index of images was proposed by Karthikeyan et al., [2006] to assess the quality degradation due to segmentation. In the interactive environment, the designer can set up various components of this quality index and ensure that the quality requirements are met by the image segments. This checking can be done online. Finally, more number of test images with a wide variety of image feature density, feature redundancy and structure needs to be segmented to study the efficacy of the progressive designer interaction.

Part (b) – Shape Optimization of the Finned Heat Exchanger

Human-guided shape optimization task was performed to obtain an optimum fin profile. It is essential to obtain an optimum profile to achieve maximum heat transfer through the finned dissipater. The results presented in this section are centered on two major areas, the progress of evolutionary algorithms in terms of fitness values, and evolution time per optimization run. To study the efficacy of the progressive interaction scheme four fin profiles (degree 1 through 4) were studied under three different conditions, viz., random optimization runs, designer interaction runs, and human subject runs. The problem parameters and operational constraints remained the same for all the three run types. The human subjects were used to study the repeatability of the performance of the progressive interaction scheme and to demonstrate the ease to understand the problem even by a beginner. A total of 4 human subjects were used for this work. Two senior level graduate students, out of which one has good understanding on the thermo-fluids area and the other one with minimal exposure to this subject. Two entry level graduates students, out of which one has a good grasp this subject matter were used in the human subjects' experiments.

Simple evolutionary algorithms were used to optimize profiles degrees 1 through 4. The fin and the operating fluid were assumed to be aluminum and water respectively. The ratio of their thermal conductivities i.e. k_s / k_f was taken to be 300 with $k_f = 0.6 W/m-K$ and $k_s = 180 W/m-K$. For each fin profile, a total of 10 evolutionary runs each were performed in random and designer interactive optimization. In addition, 8 runs per fin profile degree (2 runs per subject per profile) were conducted using human subjects' interaction. For each evolutionary run a maximum 150 mating events were done. The other stopping criterion is when the normalized fitness value is equal to or greater than 0.9. The population size for both the random and interactive runs were chosen to be 32. For the interactive population runs, nearly 10% of the population, say 3 solutions out of 32 were designer/human subjects developed solutions. This chapter presents the results of a finned heat exchanger shape optimization problem using continued human guidance. The use of an interactive population in most cases permitted the algorithm to converge more quickly to an acceptable solution whereas a random choice often got stuck in local optima. The designer's choice of “smart” initial population plays a significant role in speeding progress towards a high quality solution. Figs. 9.10 to 9.13 compares the average fitness of random population, designer generated population, and human subjects population. The data of these plots are presented in the Appendix.

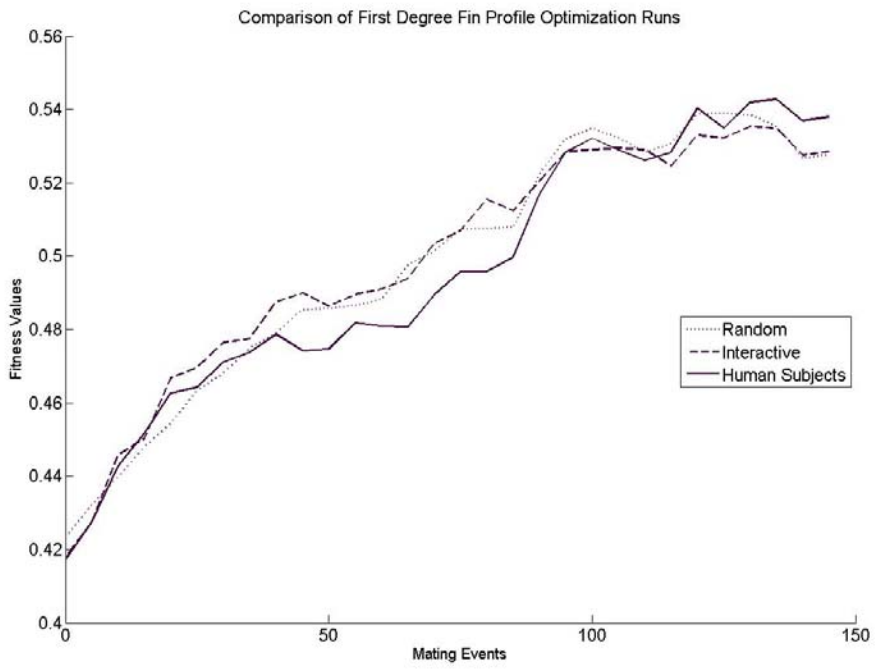


Fig. 9.10 Results of the Evolutionary Runs on the First Degree Fin Profile

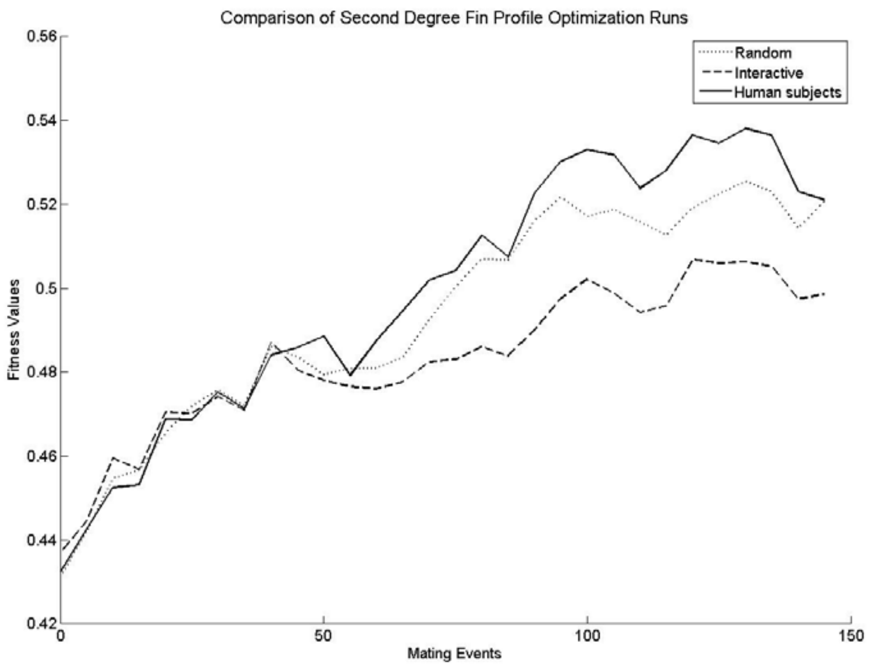


Fig. 9.11 Results of the Evolutionary Runs on the Second Degree Fin Profile

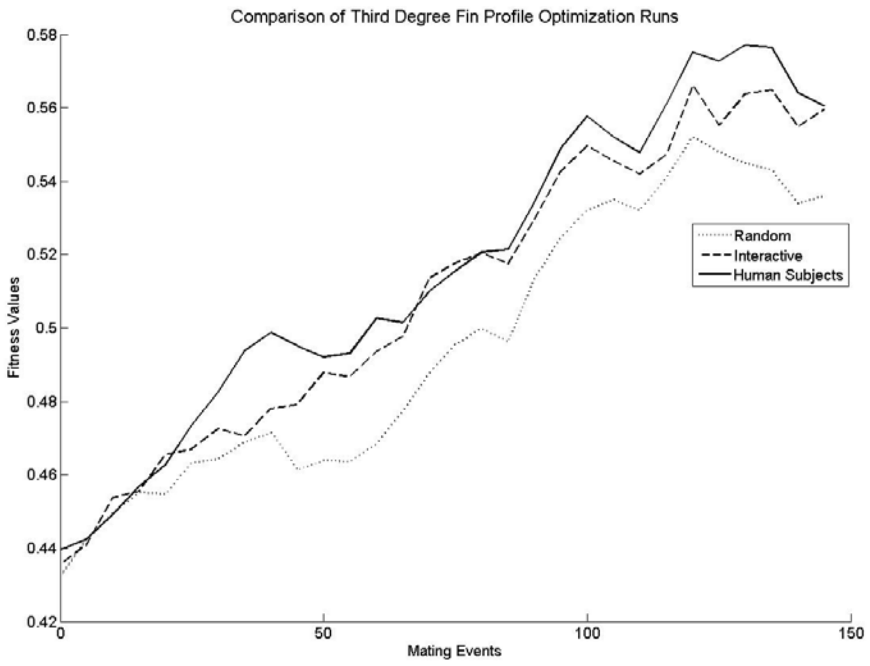


Fig. 9.12 Results of the Evolutionary Runs on the Third Degree Fin Profile

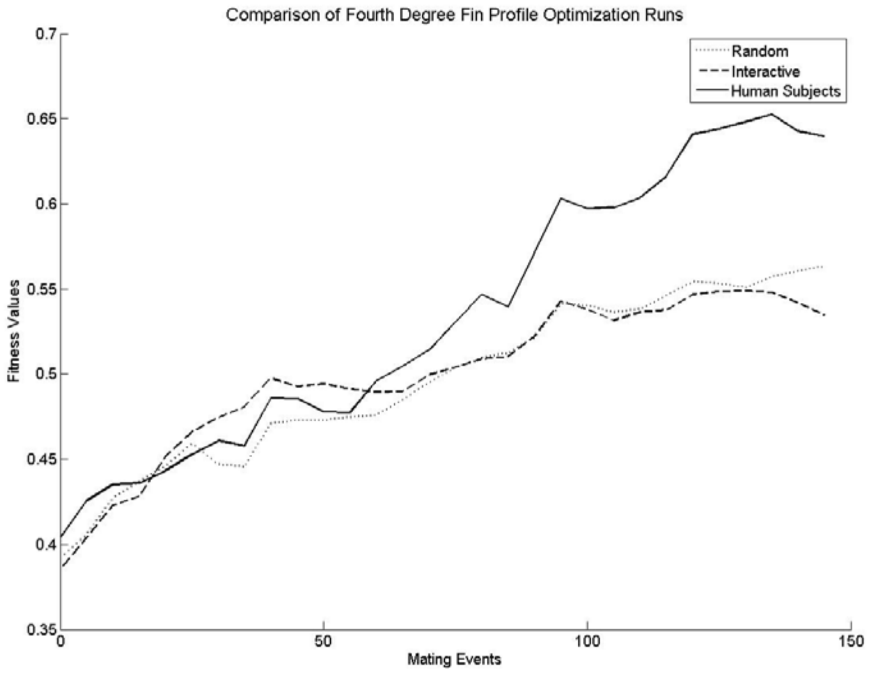
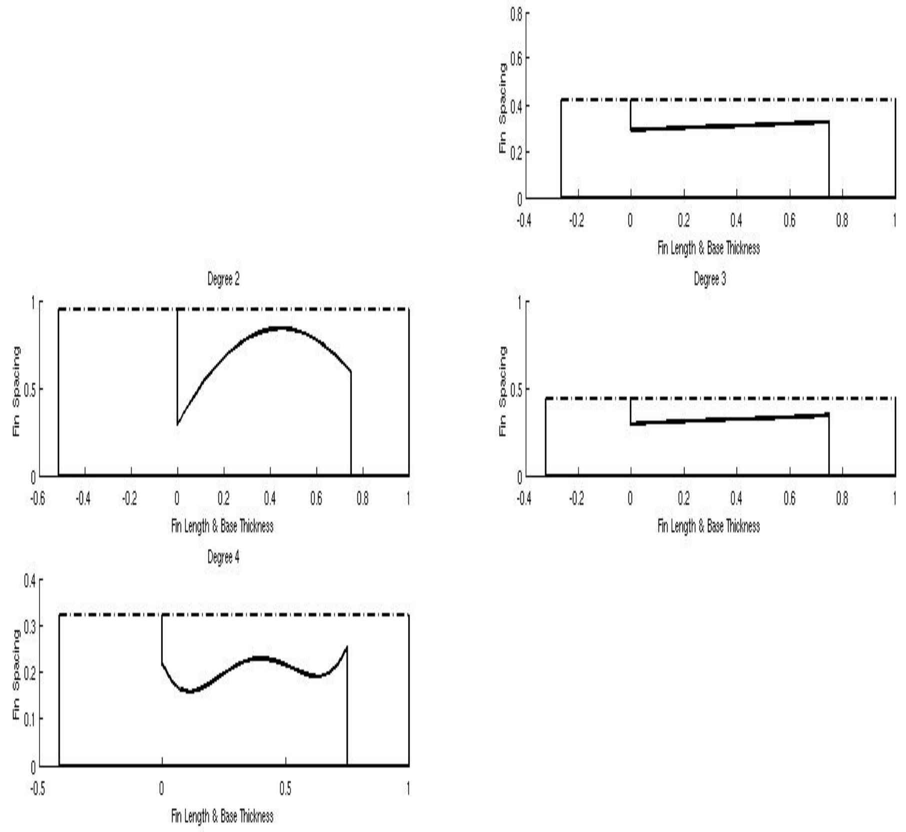


Fig. 9.13 Results of the Evolutionary Runs on the Fourth Degree Fin Profile

In all fin profiles shown in the Figs. 9.10 to 9.13, the average fitness values of the designer or human subjects' interactive populations were clearly ahead of those in the random populations. This indicates better heat transfer performance of the fin profiles generated by interaction. In addition, it is worth noting that the average initial fitness of all the random, designer interaction, and human subjects' interactions runs are all comparable. As the mating events occur, the fitness of the interactive runs shot up due to careful insertion of high performance solutions and rejection of low performing solution by the user. Thus the efficacy of the progressive designer interaction in terms of progress of fitness values is demonstrated through these Figs. The best fin profiles from each degree from a random, designer-created, and human subjects' generated population are presented in Figs. 9.14 – 9.16 respectively.

All the dimensions are normalized [0,1] range. The fin base thickness ranging from 0 to 1 is represented in the negative x-axis. The fin spacing also in the range 0 to 1 is represented along the y-axis. The fin length with a constant value of 0.75 is represented in the positive x-axis. From Figures 9.14-9.16 it is evident that in most cases, the fin profiles obtained from the both of the interactive techniques are smoother than that from the random population. This indicates higher implementation feasibility of the interactive population due to the profile simplicity. To explain the profile simplicity and its implementation friendliness Fig. 9.17 shows the temperature and velocity distribution on a 3^{rd} degree fin profile that was generated using designer interaction.



9.14 Best Fin Profiles for degree 1 to 4 using Random Population

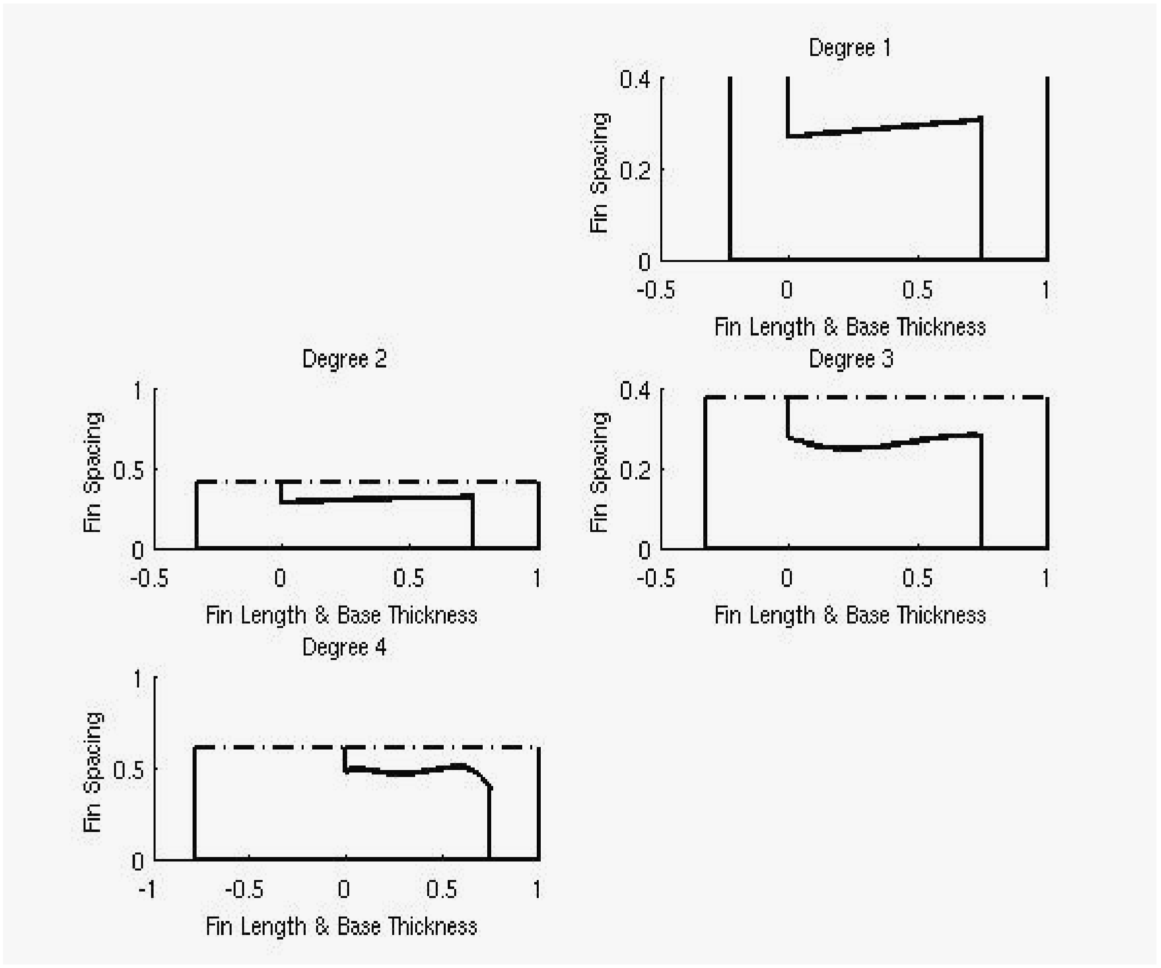


Fig. 9.15 Best Fin Profiles for degree 1 to 4 using Designer Interactive Population

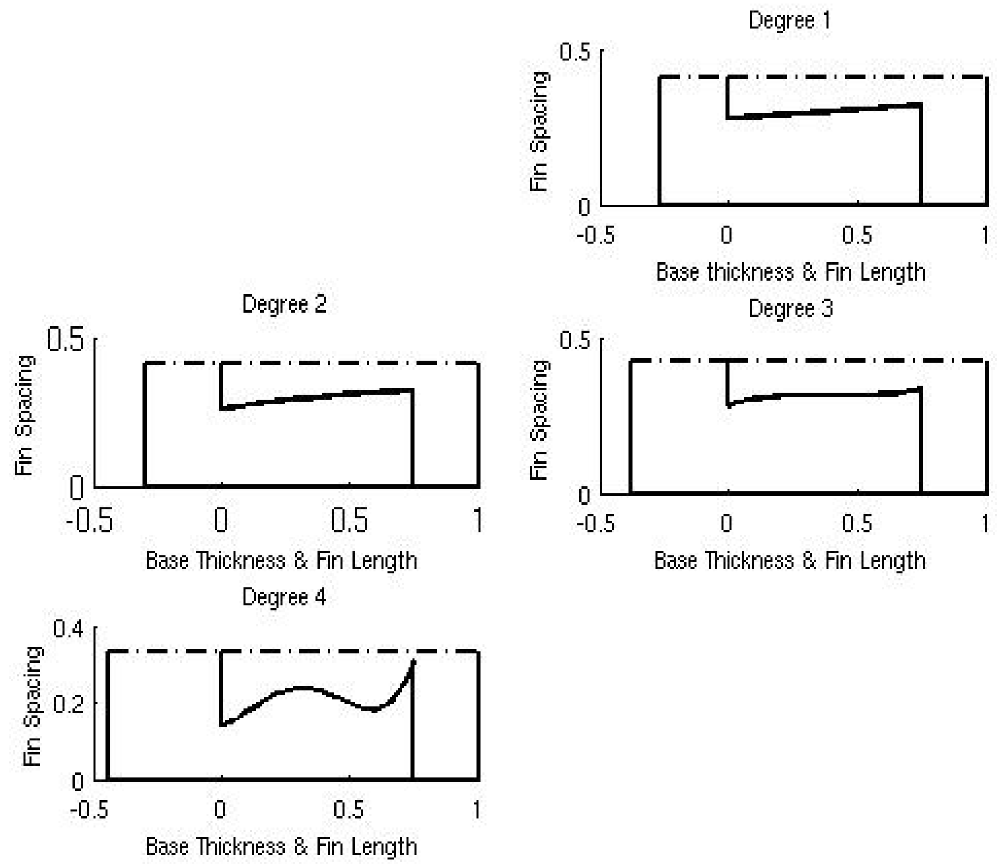
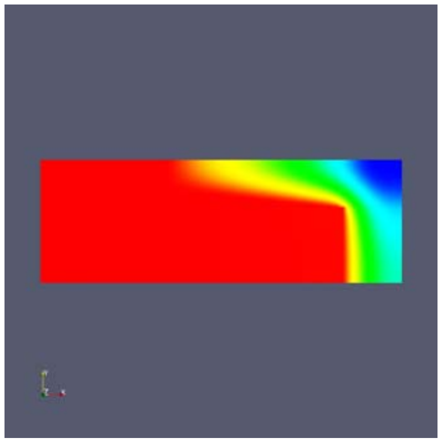
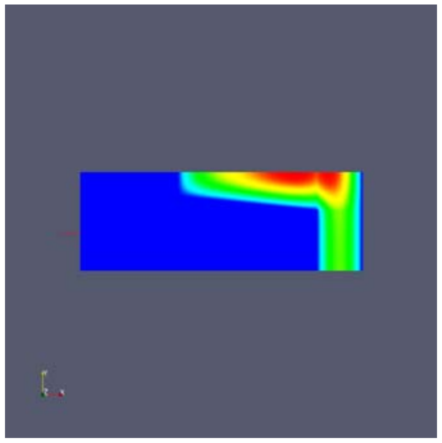


Fig. 9.16 Best Fin Profiles for degree 1 to 4 using Human Subjects' Interaction



(a) Temperature Distribution

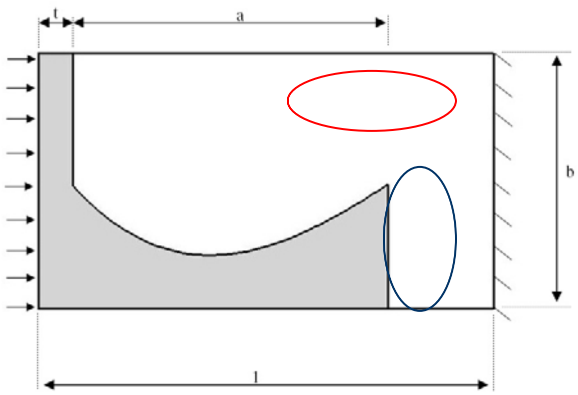


(b) Velocity Distribution

Fig. 9.17 Scalar Parameter Distribution in Designer Generated 3rd Degree Fin Profile

The second major of investigation in this test case is on time to complete an evolutionary run. This check was not done on the image segmentation and optimization test case presented in Part (a), because that problem involves varying gene length of the solution. In addition, to reduce the image segment variance, the first choice of the designer is to increase the number of segments, thereby increasing the computation time. In this test case, that difficulty doesn't exist as this test case involves fixed length of the gene through out the mating events. Table 9.2 shows the average time required to complete one evolutionary run in case of random, and both interactive evolutionary optimization processes. The time to evolution includes the calls to the CFD solver, creation of visualization toolkit (VTK) files for the solutions whenever a fitness evaluation routine is called and the time to accept the designers' choice of "smart" solution insertion during run time in case of designer interaction or human subjects' interaction.

Discussion on fin profiles. The most common fin profiles taper down. In this research, the fin profiles are not designed to taper down. When the common fin profiles are chosen for this work, the maximum velocity on the fluid occurs in the region outside of the fin (circled in upper right side), as shown in Fig. 9.18. This does not help in heat dissipation from the curved surface as best heat dissipation occurs when the maximum velocity is closed to the curved surface. In addition, the fin spacing comes into the picture, wherein the spacing cannot be too close because in that case the maximum velocity in the fluid occurs in the region (circled in lower right side) above. This is because of the no-slip conditions along the solid-fluid interfaces. Thus, once the fin-spacing is set all the points along the curved surface (y coordinates) have to lie with a certain specified distance of the fin spacing. The combination of the above two physical phenomena contribute to the fact that the fin profiles obtained are not tapering downwards, but are trying to form a “cavity” where the fluid can flow between the fins.



9.18 Description of Maximum Velocity Region

Table 9.2 Average Time to Complete One Evolutionary Optimization Run

Fin Profile	Random, min	Interactive, min	Human Subjects, min
Deg. 1	372	350	438
Deg. 2	414	314	459
Deg. 3	463	283	443
Deg. 4	532	527	525

From Table 9.2 it is understood that the designer interactive runs were helpful to expedite the solution towards the global optima. The only exclusion was the fourth degree fin profile where the average time taken by the random and the designer interactive and the human subjects’ interaction values were comparable. Nearly 25-35% time reduction was achieved due to designer interaction for the 2nd and 3rd degree fin profiles. However, the human subjects’ interaction results were not very encouraging. This is due to the wrong choice of the initial population, resulting in bad grid generation. The solver is set up in such a way that if it can't create the grid with the given fin profile configuration then it has to undergo certain mutation operation and revisit the grid generation and the solver. This is the reason why the human subjects evolutionary optimization runs were by-and-large slower when compared to the other two types.

Conclusion and Recommended Work in this Test Case

This part of the chapter presented the results available from the implementation of human-guided, progressive interaction scheme. The results establish the importance of designer interaction especially in a complex problem involving CFD analysis within optimization. Understanding the problem complexity before inserting the solutions and visualizing the preliminary results using the solutions that the designer intends to insert in the population is very important. On doing the preliminary study, the complex iterations due to the grid generation issues can be sorted out.

The recommendations for future work includes: conducting more number of evolutionary optimization runs with 10%, 20%, 30%, 40% and 50% interactive population. This study will act as a starting place to have the optimum number of designer created solution in the population. To study the model reduction feasibility in order to reduce the solver time. As a graphical visualization enhancement, the designer controlled VTK files should be loaded and visualized in the graphical plugin.

Part (c) – Interactive Nozzle Design

The results presented in this section are centered on developing the initial population with high initial fitness values. The goal of this test case is to generated higher initial fitness by interaction that would guide evolutionary algorithms to determine optimum solutions much faster, as they proceed. As presented in the other two test cases, a population size of 32 is used in this test case. For the random population all the 32 nozzle designs were generated at random. For the designer interactive population and human subjects' population approximately 10% (i.e., 3 solutions) were created by the interaction. A total of 5 evolutionary runs were conducted on random population, designer interactive population, and human subjects' interactive population. The comparison of the fitness values from these runs are presented in Fig. 9.19.

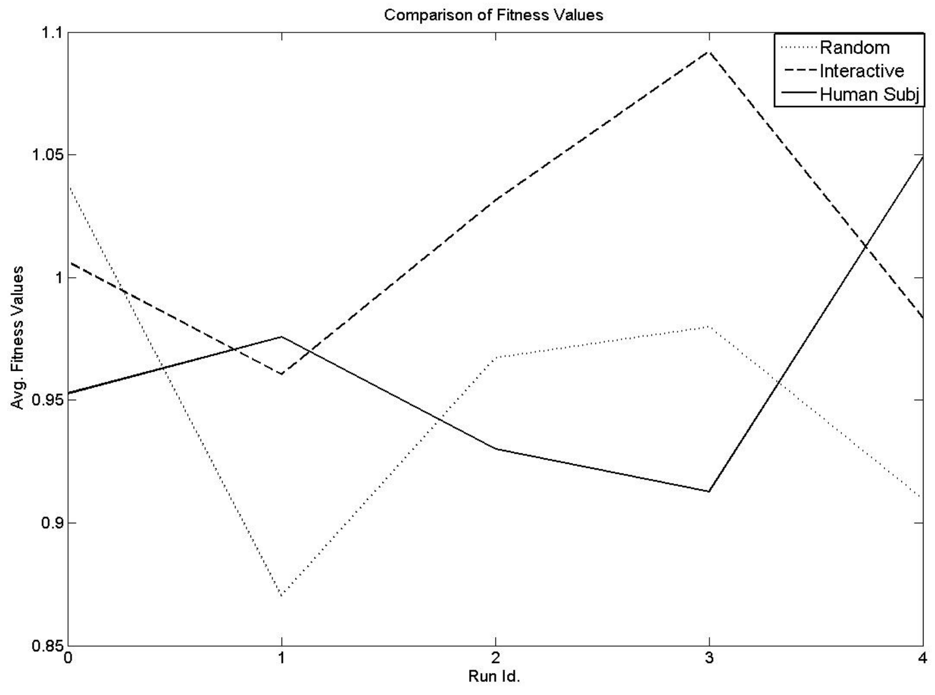


Fig. 9.19 Comparison of Initial Fitness Values of Nozzle Population

From Fig. 9.19 it is evident that the average fitness values of the designer generated population is higher than that of the random population. The proof-of-the-concept results presented in Fig. 9.19 clearly indicates that the designer interaction can create solutions that can guide the evolutionary algorithms towards optimum solutions.

Conclusion and Recommendations for Future Work

The prime goal of this test case was to integrate the progressive interaction framework with a commercial high fidelity CFD solver. This goal was achieved and was demonstrated using the initial results available. Many design parameters were successfully inserted to the computational unit during run-time. And the designer interaction increased the fitness values and hence, the likelihood to progress towards to optima faster. The future work in this test case include: run more number of runs to establish that the designer with significant amount of a priori knowledge can always generate a population of nozzles with higher initial fitness. In addition, evolutionary optimization runs with 20%, 30%, 40% and 50% interactive population can be conducted. The results obtained from the evolutionary mating events can also be recorded.

Appendix

A.1 Example Setup File for the Nozzle Project

```
#!/bin/csh -f
cd ./nozzle4/run_0
source /usr/local/starcd/etc/setstar

proam << EOF
x
star
n
n

memory maxvrt 800000
memory maxcel 600000
memory maxncp 8000
memory maxsc2 20000000
memory maxncy 7000
memory maxnbu 200000
v,1,0,0,0
v,2,0,0,36
v,3,5,0,36
v,4,5,0,0
v,5,0.15,0,0
v,6,0.15,0,5
v,7,0.35,0,5
v,8,0.35,0,0
v,9,0.15,0,6.06
v,10,0,0,6.06
v,11,0,0,12.06
v,12,5,0,12.06
v,13,5,0,6.06
v,14,5,0,5
v,15,1.206,0,36
v,16,1.875,0,5
v,17,1.875,0,6.06
v,18,0.99,0,5
v,19,0.99,0,6.06
v,20,1.07,0,6.18245
v,21,1.15,0,6.3049
v,22,1.198,0,6.42735
v,23,1.214,0,6.5498
v,24,1.27,0,6.67224
v,25,1.342,0,6.79469
v,26,1.422,0,6.91714
v,27,1.422,0,7.03959
v,28,1.422,0,7.16204
v,29,1.478,0,7.28449
v,30,1.494,0,7.40694
v,31,1.494,0,7.52939
v,32,1.494,0,7.65184
```

```
v,33,1.494,0,7.77428
v,34,1.494,0,7.89673
v,35,1.494,0,8.01918
v,36,1.494,0,8.14163
v,37,1.494,0,8.26408
v,38,1.494,0,8.38653
v,39,1.494,0,8.50898
v,40,1.494,0,8.63143
v,41,1.494,0,8.75388
v,42,1.494,0,8.87633
v,43,1.494,0,8.99877
v,44,1.494,0,9.12122
v,45,1.494,0,9.24367
v,46,1.494,0,9.36612
v,47,1.494,0,9.48857
v,48,1.494,0,9.61102
v,49,1.494,0,9.73347
v,50,1.494,0,9.85592
v,51,1.494,0,9.97837
v,52,1.494,0,10.1008
v,53,1.494,0,10.2233
v,54,1.494,0,10.3457
v,55,1.494,0,10.4682
v,56,1.494,0,10.5906
v,57,1.494,0,10.7131
v,58,1.494,0,10.8355
v,59,1.494,0,10.958
v,60,1.494,0,11.0804
v,61,1.494,0,11.2029
v,62,1.494,0,11.3253
v,63,1.494,0,11.4478
v,64,1.414,0,11.5702
v,65,1.358,0,11.6927
v,66,1.302,0,11.8151
v,67,1.286,0,11.9375
v,68,1.206,0,12.06
v,69,1.19,0,6.06
v,70,1.27,0,6.18245
v,71,1.35,0,6.3049
v,72,1.398,0,6.42735
v,73,1.414,0,6.5498
v,74,1.47,0,6.67224
v,75,1.542,0,6.79469
v,76,1.622,0,6.91714
v,77,1.622,0,7.03959
v,78,1.622,0,7.16204
v,79,1.678,0,7.28449
v,80,1.694,0,7.40694
v,81,1.694,0,7.52939
v,82,1.694,0,7.65184
v,83,1.694,0,7.77428
v,84,1.694,0,7.89673
v,85,1.694,0,8.01918
```

```
v,86,1.694,0,8.14163
v,87,1.694,0,8.26408
v,88,1.694,0,8.38653
v,89,1.694,0,8.50898
v,90,1.694,0,8.63143
v,91,1.694,0,8.75388
v,92,1.694,0,8.87633
v,93,1.694,0,8.99877
v,94,1.694,0,9.12122
v,95,1.694,0,9.24367
v,96,1.694,0,9.36612
v,97,1.694,0,9.48857
v,98,1.694,0,9.61102
v,99,1.694,0,9.73347
v,100,1.694,0,9.85592
v,101,1.694,0,9.97837
v,102,1.694,0,10.1008
v,103,1.694,0,10.2233
v,104,1.694,0,10.3457
v,105,1.694,0,10.4682
v,106,1.694,0,10.5906
v,107,1.694,0,10.7131
v,108,1.694,0,10.8355
v,109,1.694,0,10.958
v,110,1.694,0,11.0804
v,111,1.694,0,11.2029
v,112,1.694,0,11.3253
v,113,1.694,0,11.4478
v,114,1.614,0,11.5702
v,115,1.558,0,11.6927
v,116,1.502,0,11.8151
v,117,1.486,0,11.9375
v,118,1.406,0,12.06
spl,1,vran,19,68,1
spl,2,vran,69,118,1
ctab,4,shel,6
patc,8,7,14,4,53,49
patc,1,10,9,5,260,6
patc,16,17,13,14,15,44
patc,69,118,12,13,64,45
patc,10,11,68,19,128,25
patc,68,15,3,12,256,40
patc,11,2,15,68,513,25
patc,6,9,19,18,22,18
patc,18,19,17,16,22,18
csys,2
ctyp,1
cset,all
vcex,1,,cset,,,local,0,5.34135,0,both,uniform
csys,1
vset all
vmerge vset
c
[illegible]
```

```
cset news flui
cptable,2,arbit,1,2,4,3,on,0.02
cptype,2
cpcreate cset
cpcheck,,all,,,newset,,fix
cpdel cpset
cset,news,flui
check,cset,,negvol,,newset
cdel,cset
pmat,1,fluid,H2O
dens,cons,997.56
spec,cons,4181.72
lvis,cons,0.0008887
cond,cons,0.62027
turb,ke,1,stan
lowre,off
coke,0.09,1.44,1.92,1.44,-0.33,0.419,1,1.219,0.9,,,
rdef,2,inlet,standard
0,0,18.1386,1,0,997.56
rturb,2,mixlen,0.1,0.0033
rinlet,2,mass,n
*set region,2
*set rmax,0.15
*set rmin,0
*set z,0
cset news flui
live surf create
*get s1,mxct
cset news type s1
cset subs grange,rmin - 0.0001,rmax + 0.0001,,,z - 0.0001, z +
0.0001,2
bshell,region - s1,cset
rdef,3,pressure,standard
piezo,0,n,n,n
rturb,3,zgrad
rtpress,3,temp,,conc
*set region,3
*set rmax,5
*set rmin,0
*set z,36
cset news type s1
cset subs grange,rmin - 0.0001,rmax + 0.0001,,,z - 0.0001, z +
0.0001,2
bshell,region - s1,cset
rdef,4,pressure,standard
piezo,0,n,n,n
rturb,4,zgrad
rtpress,4,temp,,conc
*set region,4
*set rmax,5
*set rmin,0.35
*set z,0
cset news type s1
[illegible]
```

```
cset subs grange,rmin - 0.0001,rmax + 0.0001,,,z - 0.0001, z +
0.0001,2
bshell,region - s1,cset
rdef,8,symplane,standard
*set region,8
cset news type s1
cset subs grange,,,-0.0001,0.0001,,,1
bshell,region - s1,cset
rdef,9,symplane,standard
*set region,9
cset news type s1
csys,1
local,4,cart,0,0,0,5.34135,0,0,
csys,4
cset subs grange,,,-0.0001,0.0001,,,4
bshell,region - s1,cset
rdef,10,wall,standard
slip,standard,,
*set region,10
cset news type s1
cset subs grange,4.9999,5.0001,,,,,2
bshell,region - s1,cset
csys,1
moni,52131
pres,100000,52131
tdatum,273
iter,1000,0.001
relax,rlvel,0.7
relax,rlp,0.15
relax,rlke,0.7
relax,rlvis,1
wdata,restart,25,0,star.pst
cset news flui
cset dele grange,,,,,4.9999,6.0601,2
cset dele grange,,,,,12.0599,13.06,2
live surf create
*get s2,mxct
cset news type s2
cset dele grange,-0.0001,0.1501,,,4.9,5.1,2
cset dele grange,-0.0001,1.2061,,,12.0599,12.0601,2
cset dele grange,,,,,35.9999,36.0001,2
cdel cset
cset news type s2
ctab,s2 + 1,shell,18,,,,,,,off,,light
cset subs grange,-0.0001,1.2061,,,12.0599,12.0601,2
ctyp,s2 + 1
cmod,cset
cset news type s2
ctab,s2 + 2,shell,19,,,,,,,off,,light
cset subs grange,,,,,35.9999,36.0001,2
ctyp,s2 + 2
cmod,cset
ctab,20,fluid,22,,,,,,,off,,light
```

```
ctyp,20
cset news type s2
cset dele grange,0.1499,0.1501,,,0,6.06,2
cset dele grange,,,-0.0001,0.0001,,,2
cset dele grange,,,-0.0001,0.0001,,,4
cset,add,neighbor,cset
cset,dele,shell
cset,dele,grange,,,,,4.9999,1e6,2
cmod,cset
ctab,30,fluid,32,,,,,,,off,,light
ctyp,30
cset news type s2
cset dele grange,0.1499,0.1501,,,0,6.06,2
cset dele grange,,,-0.0001,0.0001,,,2
cset dele grange,,,-0.0001,0.0001,,,4
cset,add,neighbor,cset
cset,dele,shell
cset,dele,type,20
cmod,cset
ctab,21,fluid,23,,,,,,,off,,light
ctyp,21
cset,news,type s2 + 1
cset,add,neighbor,cset
cset,dele,shell
cset,dele,grange,,,,,12.0599,1e6,2
cmod,cset
ctab,31,fluid,33,,,,,,,off,,light
ctyp,31
cset,news,type s2 + 1
cset,add,neighbor,cset
cset,dele,shell
cset,dele,grange,,,,,0,12.0601,2
cset,dele,grange,1.2059,1e6,,,,,2
cmod,cset
ctab,22,fluid,24,,,,,,,off,,light
ctyp,22
cset,news,type s2 + 2
cset,add,neighbor,cset
cset,dele,shell
cmod,cset
prfield,none,,,user
geom,star.ccm,0.0254,ccm,nobackup
prob,star.prob
save,star.mdl
quit,nosave
EOF
cd ../../
```

A 2. Fitness Values of First Degree Fin Profile

Mating Event Number	Average Fitness Value of the Population		
	Random population	Designer population	Human subjects population
0	0.423222	0.418055	0.417213
5	0.432098	0.42756	0.427465
10	0.439811	0.445821	0.442717
15	0.448007	0.450292	0.451695
20	0.454412	0.466704	0.462633
25	0.463237	0.469594	0.464182
30	0.467993	0.476451	0.471066
35	0.474878	0.477482	0.473765
40	0.478934	0.487503	0.478647
45	0.485431	0.490016	0.474175
50	0.485731	0.486484	0.474562
55	0.486611	0.489542	0.481783
60	0.488104	0.490949	0.480968
65	0.49747	0.493896	0.480719
70	0.501178	0.503501	0.489339
75	0.507308	0.507047	0.495719
80	0.507581	0.515517	0.495812
85	0.507931	0.512311	0.499705
90	0.522248	0.520392	0.516786
95	0.53213	0.528421	0.528399
100	0.534829	0.529072	0.532166
105	0.532343	0.529513	0.528947
110	0.528315	0.529058	0.526141
115	0.530789	0.524518	0.528316
120	0.538729	0.532954	0.540358
125	0.53894	0.532173	0.534921
130	0.53859	0.535409	0.541958
135	0.535336	0.534789	0.542792
140	0.526648	0.527486	0.536869
145	0.527724	0.528494	0.537998

A 3. Fitness Values of Second Degree Fin Profile

Mating Event Number	Average Fitness Value of the Population		
	Random population	Designer population	Human subjects population
0	0.431077	0.436878	0.432276
5	0.441984	0.444481	0.442614
10	0.454714	0.45953	0.452543
15	0.456748	0.456892	0.453194
20	0.465545	0.470492	0.468832
25	0.471842	0.470217	0.468678
30	0.475725	0.474142	0.475212
35	0.472097	0.470993	0.471354
40	0.48599	0.487125	0.484061
45	0.48365	0.480557	0.485874
50	0.479419	0.477986	0.488488
55	0.480926	0.476608	0.479143
60	0.480991	0.475971	0.48756
65	0.483523	0.477619	0.494633
70	0.492393	0.482423	0.501853
75	0.500339	0.483076	0.504199
80	0.506906	0.486016	0.512625
85	0.506807	0.483878	0.507386
90	0.51598	0.490007	0.522579
95	0.521685	0.497477	0.530265
100	0.517199	0.502172	0.533056
105	0.518703	0.498831	0.53176
110	0.515843	0.494275	0.523769
115	0.512628	0.495818	0.528085
120	0.519165	0.506888	0.536415
125	0.522627	0.505953	0.534544
130	0.525474	0.50629	0.538092
135	0.522872	0.505215	0.536419
140	0.514316	0.497386	0.523101
145	0.520706	0.49857	0.521116

A 4 Fitness Values of Third Degree Fin Profile

Mating Event Number	Average Fitness Value of the Population		
	Random population	Designer population	Human subjects population
0	0.432163	0.43556	0.439559
5	0.442525	0.441002	0.442386
10	0.449598	0.453785	0.449168
15	0.455166	0.455477	0.456854
20	0.454711	0.465557	0.462746
25	0.463289	0.467018	0.473539
30	0.464259	0.472578	0.48252
35	0.468904	0.470555	0.493657
40	0.471572	0.477965	0.498714
45	0.461358	0.479129	0.495005
50	0.463924	0.487821	0.492112
55	0.463583	0.486659	0.493099
60	0.468354	0.493569	0.502676
65	0.477308	0.497628	0.501455
70	0.487647	0.513496	0.509837
75	0.495555	0.517806	0.515613
80	0.499872	0.520328	0.52071
85	0.496218	0.517512	0.521394
90	0.513405	0.529626	0.534251
95	0.524634	0.542726	0.548988
100	0.532048	0.549598	0.557706
105	0.535099	0.545492	0.552053
110	0.532	0.541917	0.547742
115	0.540742	0.547268	0.560956
120	0.552081	0.566073	0.575203
125	0.54797	0.555224	0.572726
130	0.544946	0.56379	0.57706
135	0.543107	0.564934	0.576484
140	0.533884	0.554819	0.564021
145	0.535966	0.55957	0.560655

A 5 Fitness Values of Fourth Degree Fin Profile

Mating Event Number	Average Fitness Value of the Population		
	Random population	Designer population	Human subjects population
0	0.391701	0.385164	0.403753
5	0.405872	0.403862	0.425562
10	0.427271	0.423035	0.435079
15	0.436759	0.428253	0.436057
20	0.44597	0.451445	0.443198
25	0.459123	0.466051	0.452766
30	0.446977	0.474707	0.460938
35	0.445508	0.48094	0.457703
40	0.470816	0.497587	0.485891
45	0.473035	0.492731	0.485409
50	0.473024	0.49437	0.477965
55	0.474757	0.491195	0.477346
60	0.475919	0.489656	0.49622
65	0.48498	0.489679	0.504723
70	0.495107	0.4995	0.514144
75	0.503981	0.504241	0.530665
80	0.509556	0.508835	0.54677
85	0.512428	0.510375	0.539464
90	0.521384	0.522313	0.571226
95	0.541434	0.542772	0.60323
100	0.540508	0.53799	0.597235
105	0.536409	0.531671	0.597962
110	0.538147	0.536319	0.603615
115	0.546445	0.537753	0.616146
120	0.55445	0.546867	0.640868
125	0.553212	0.548447	0.643955
130	0.550825	0.549026	0.647985
135	0.557118	0.548058	0.652571
140	0.560397	0.54183	0.642769
145	0.563253	0.534855	0.639796

References

ABET, (1995), "Criteria for Accrediting Programs in Engineering in the United States," *1995-96 Accreditation Cycle, Accreditation Board in Engineering and Technology,* Inc., Baltimore, MD.

Anderson, D.E., Lesh, N., Marks, J., Mirtich, B., Ratajczak, D., and Ryall, K., (2000), "Human-Guided Simple Search," *In Proceedings of AAAI 2000.*

Arora, J.S., (1989), "Introduction to Optimum Design," *Mc-Graw Hill Book Company*.

Ashlock, D., Walker, J., and Smucker, M., (1999), "Graph Based Genetic Algorithms," *Proceedings of the Congress on Evolutionary Computation, IEEE Press,* vol. 2, pp. 1362-1368.

Ashlock, D.A., Bryden, K.M., Johnson, P.E., and McCorkle, D.S., (2005), "A Data Segregation Strategy Using Graph Based Evolutionary Algorithms," Accepted for Publication in the *International Journal of General Systems*.

Ashlock, D.A., (2006), "Evolutionary Computation for Modeling and Optimization," *Springer*.

Ashlock, D.A.., Karthikeyan, B., and Bryden, K.M., (2006), "Non-photorealistic Rendering of Images as Evolutionary Stained Glass," *Proceedings of the IEEE Congress on Evolutionary Computation 2006*, pp. 7440-7447, July 16-21 2006, Vancouver, BC, Canada.

Asimov, M., (1962), "Introduction to Design," *Prentice-Hall,* Englewood Cliffs, NJ.

Aukstakalnis, S., and Blatner, D., (1992), "Silicon Mirage: The Art and Science of Virtual Reality," *Peachpit Press,* Berkeley.

Back, T., (1996), "Evolutionary Algorithms in Theory and Practice," *Oxford University Press,* New York.

Beazley, D.M., and Lomdahl, P.S., (1996), "Lightweight Computational Steering of Very Large Scale Molecular Dynamics Simulations," *Presented at SuperComputing,* Pittsburgh, PA.

Bellman, R.E., (1957), "Dynamic Programming," *Princeton University Press*, Princeton, NJ.

Birmingham, R., Cleland, G., Driver, R., and Maffin, D., (1997), "Understanding Engineering Design," *Prentice Hall.*

Bloebaum, C.L., Hajela, P., and Sobieszczanski-Sobieski, J., (1992), "Non-hierarchic System Decomposition in Structural Optimization," *Engineering Optimization,* vol. 19, pp. 171-186.

Blumrich, J.F., (1970), *Science,* vol. 168, pp. 1551-1554.

Branke, J., Kauβler, T., and Schmeck, H., (2000), "Guiding Multi Objective Evolutionary Algorithms Towards Interesting Regions," *Technical Report TR No. 399, Institute AIFB, University of Karlsruhe,* Germany.

Branke, J., Kauβler, T., and Schmeck, H., (2001), "Guidance in Evolutionary Multi-objective Optimization," *Advances in Engineering Software,* vol. 32, pp. 499-507.

Branke, J., Deb, K., Dierolf, H., and Osswald, Matthias., (2004), "Finding Knees in Multi-Objective Optimization," *KanGAL Report Number 2004010.*

Bryden, K.M., Ashlock D.A., McCorkle, D.S., and Urban, G.L., (2003), "Optimization of Heat Transfer Utilizing Graph Based Evolutionary Algorithms," *International Journal of Heat and Fluid Flow,* vol. 24 (2), pp. 267-277.

Bryden, K.M., Ashlock, D.A., and Corns, S.M., (2006), "Graph Based Evolutionary Algorithms," IEEE Transactions on Evolutionary Computation, Accepted.

Buhl, H.R., (1960), "Creative Engineering Design," *Iowa State University Press.*

Cain, W.D., (1969), "Engineering Product Design," *London Business Books Limited.*

Cantu-Paz, E., (2000), "Markov Chain Models of Parallel Genetic Algorithms," *IEEE Transactions on Evolutionary Computation,* vol. 4 (3), pp. 216-226, 2000.

Chandrasekharan, (1989), "A Framework for Design Problem-Solving," WAID 89, *Workshop on Research Directions for AI in Design.*

Chen, W., Allen, J.K., Marvis, D., and Mistree F., (1996), "A Concept Exploration Method for Determining Robust Top-level Specifications," *Engineering Optimization,* vol. 26 (2), pp. 137-158.

Chin-Hsien, L., Chin-Hsiang, C., and Chun-Yin, W., (2001), "Shape Design for Heat Conduction Problems Using Curvilinear Grid Generation, Conjugate Gradient, and Redistribution Methods," *Numerical Heat Transfer - Part A,* vol. 39, pp. 487-510.

Chimani, M., Lesh, N., Mitzenmacher, M., Sidner, C., and Tanaka, Hidetoshi, (2004), "A Case Study in Large Scale Interactive Optimization," *Mitsubishi Electric Research Laboratories, TR2004-113.*

Coello Coello, C.A., Van Veldhuizen, D.A., and Lamont, G.B., (2002), "Evolutionary Algorithms for Solving Multi-Objective Problems," *Kluwer Academic Publishers.*

Collins, T.D., (1998), "Understanding Evolutionary Computer: A Hands-on Approach," in *WCCI-98.*

Cooper, R.G., (1984), "The Performance Impact of Product Innovation Strategies," *European Journal of Marketing*, vol. 18 (5), pp. 5-54.

Coyne, R.D., Rosenman, M.A., Radford, A.D., Balachandran, M., and Gero, J.S., (1990), "Knowledge Based Design System," *Addison-Wesley,* Reading, MA.

Cross, N., (1989), "Engineering Design Method," *John Wiley & Sons,* Chichester.

Cvetkovic, D., and Parmee, I.C., (1999), "Designer's Preferences and Multi-Objective Preliminary Design Process," *In: Banzhaf, W., Editor, Genetic and Evolutionary Computation Conference Proceedings,* vol. 2, pp. 1504-1509.

Cvetkovic, D., and Parmee, I.C., (2000), "Use of Preferences for GA-based Multi-Objective Optimization," *In: Parmee, I.C., (Editor), Adaptive Computing and Manufacture, Berlin: Springer,* pp. 249-260.

Dani, T.H., Chu, C.P., and Gadh, R., (1997), "COVIRDS: Shape Modeling in a Virtual Reality Environment," *1997 ASME Design Technical Conference and Computers in Engineering Conference,* Sacramento, CA, Sept. 14-17.

Deb, K., (1999), "Solving Goal Programming Problems using Multi-objective Genetic Algorithms," *Congress on Evolutionary Computation, IEEE,* vol. 1, pp. 77-84.

Deb, K., (2001), "Multi-Objective Optimization using Evolutionary Algorithms," *John Wiley and Sons.*

Deb, K and Gupta, H., (2004), “Introducing Robustness in Multi-Objective Optimization,” *KanGAL Report Number 2004016*.

Deb, K and Chaudhuri, S., (2005), “I-EMO: An Interactive Evolutionary Multi-Objective Optimization Tool,” *KanGAL Report Number 2005003*.

Deitz, D., (1995), “Real Engineering in a Virtual World,” *Mechanical Engineering,* vol. 117, pp. 78-85.

Deng, Y.M., Britton, G.A., Lam, Y.C., Tor S.B., and Ma, Y.S., (2002), “Feature-based CAD-CAE Integration Model for Injection-Molded Product Design,” *International Journal of Production Research,* vol. 40 (15), pp. 3737-3750.

Diachin, D., Freitag, L., Heath, D., Herzog, J., Michels, W., and Plassmann, P., (1996), “Collaborative Virtual Environments Used in the Design of Pollution Control Systems,” *International Journal of Supercomputer Applications and High Performance Computing,* vol. 10 pp. 223-235.

Duvurru, S., and Stephanopouls, G., (1999), “Knowledge-based System Applications in Engineering Design,” *Research at MIT, AI Magazine,* vol.10 (3), pp. 79-96.

Ellis, G., (1996), “Digital Clay: Transforming Automobile Design,” *IRIS Universe,* No. 37, pp. 28-32.

Engelbrecht, J.J., McCorkle, D.S., and Bryden, K.M., “Optimization Study of a Hydraulic Mixing Nozzle,” *To be Submitted*.

Ertas, A., and Jones, J.C., (1996), “The Engineering Design Process,” 2nd edition, *John Wiley & Sons*.

Evans, P.T., Vance, J.M., and Dark, V.J., (1999), “Assessing the Effectiveness of Traditional and Virtual Reality Interfaces in Spherical Mechanism Design,” *Journal of Mechanical Design,* vol. 121 (4), pp. 507-514.

Fabbri, G., (1997), “A genetic algorithm for fin profile optimization,” *International Journal of Heat and Mass Transfer,* 40 (9), 2165-2172.

Fabbri, G., (1998), “Optimization of Heat Transfer through Finned Dissipaters Cooled by Laminar Flow,” *International Journal of Heat and Fluid Flow,* vol. 19, pp. 644-654.

Fan, L.S., and Zhu, C., (1998), “Principles of Gas-Solid Flows,” *Cambridge University Press,* Cambridge, United Kingdom.

Fogel, L.J., (1962), “Autonomous Automata,” *Industrial Research,* vol. 4 (1), pp. 14-19.

Fogel, D.B., (1988), “An Evolutionary Approach to the Traveling Salesman Problem,” *Biological Cybernatics,* vol. 60 (2), pp. 139-144.

Fogel, L.J., (1999), “Artificial Intelligence through Simulated Evolution. Forty Years of Evolutionary Programming,” *John Wiley & Sons,* New York.

Foster, G.F. and G.S. Dulikravich, (1997), “Three-dimensional aerodynamic shape optimization and gradient search algorithms,” *Journal of Spacecraft Rockets*, 34:36-42.

French, M.J., (1985), “Conceptual Design for Engineers,” *The Design Council,* London.

Furlong, T.J., Vance, J.M., and Larochelle, P.M., (1999), “Spherical Mechanism Synthesis in Virtual Reality,” *Journal of Mechanical Design,* vol. 121 (4), pp. 515-520.

Gajda, W.J., and Biles, W.C., (1978), “Engineering Modeling and Computation,” *Houghton Mifflin Company,* Boston, 1978.

Garey, M., and Johnson, D., (1979), "Computers and Intractability: A Guide to the Theory of NP-Completeness," *Freeman.*

Geist II, G.A., J.A. Kohl, and P.M. Papadopoulos, (1997), "CUMULVS: providing fault tolerance, visualization, and steering of parallel applications," *International Journal of Supercomputer Applications and High Performance Computing*, 11:224-235.

Gero, J.S., and Schnier, T., (1995), "Evolving Representations of Design Cases and Their Use in Creative Design," *Preprints Computational Models of Creative Design,* Key Center of Design Computing, University of Sydney, pp. 343-368.

Gero, J.S., (1997), "Genetic Engineering and Design Problems," Dasgupta, D., and Michalewicz, Z., (Eds), *Evolutionary Algorithms in Engineering Applications; Springer-Verlag,* pp. 47-68.

Giachetti, R.E., (2004), "A Framework to Review the Information Integration of the Enterprise," *International Journal of Production Research,* vol. 42 (6), pp. 1147-1166.

Glover, F., and Laguna, M., (1997), "Tabu Search," *Kluwer Academic Publishers,* Boston, Massachusetts.

Goldberg, D.E., (1983), "Computer-Aided Gas Pipeline Operation Using Genetic Algorithms and Rule Learning," *Ph.D Dissertation, University of Michigan.*

Goldberg, D.E., (1989), "Genetic Algorithms in Search, Optimization and Machine Learning," *Addison-Wesley Publishing Company,* Reading, Massachusetts.

Gonzalez, R.C., and Woods, R.E., (2002), "Digital Image Processing," Second Edition, *Prentice Hall.*

Gonzalez, R.C., Woods, R.E., and Eddins, S.L., (2004), "Digital Image Processing Using Matlab," *Prentice Hall.*

Hernandez, S., and Fontan, A., (2002), "Practical Applications of Design Optimization," *WIT Press,* UK.

Hill, P.H., (1970), "The Science of Engineering Design," *Holt, Rinehart, and Winston*, New York.

Holland, J.H., (1975), "Adaptation in Natural and Artificial Systems," *University of Michigan Press,* Ann Arbor, MI.

Holt, K., (1983), "Product Innovation Management," *Butterworth*, London.

Huang, G., and Bryden, K.M., (2005), "Introducing Virtual Engineering Technology into Interactive Design Process with High-Fidelity Models," *Proceedings of the Winter Simulation Conference.*

Huang, G., (2006), "An Interactive Design Environment for Coal Piping System," Doctor *of Philosophy Dissertation, Iowa State University*

Jablonowski, D.J., J.D. Bruner, B. Bliss, and R.B. Haber, (1993), "VASE: the visualization and application steering environment," *Proceedings of Supercomputing'93*, 560-569.

Jaluria, Y., (1998), "Design and Optimization of Thermal Systems," *McGraw Hill.*

Jayaram, S., Jayaram, U., Wang, Y., Tirumali, H., Lyons, K., and Hart, P., (1999), "VADE: A Virtual Assembly Design Environment," *IEEE Computer Graphics and Applications,* vol. 19 (6), pp. 44-50.

Jones, J.C., and Thornley, D.G., (Eds), (1963), "Proceedings of the Conference on Design Methods," *Pergamon Press,* Oxford, UK.

Karthikeyan, B., Bryden, K.M., and Ashlock, D. A., (2003), “Visualizing Information Flow in Graph Based Evolving Population,” *Intelligent Engineering Systems Through Artificial Neural Networks, ANNIE 2003, ASME Press* , vol. 13, pp. 299 – 304.

Karthikeyan, B., Bryden, K.M., Ashlock, D.A., (2005), “Low-Impact Image Segmentation Using Balanced Weighted Voronoi Tessellations,” *Intelligent Engineering Systems Through Artificial Neural Networks, ANNIE 2005, ASME Press,* vol. 15, pp. 533-542.

Karthikeyan, B., Ashlock, D.A., and Bryden, K.M., “A Novel Image Segmentation Technique using Balanced Weighted Voronoi Tessellations,” *Submitted to the International Journal of General Systems (2007)..*

Kihonge, J.N., Vance, J.M., and Larochelle, P.M., (2002), “Spatial Mechanism Design in Virtual Reality with Networking,” *Journal of Mechanical Design,* vol. 124 (3), pp. 435-440.

Kirkpatrick, S., Gellatt, C., and Vecchi, M., (1983), “Optimization by Simulated Annealing,” *Science, 220(4598),* pp. 671-680.

Klau, G.W., Lesh, N., Marks, J., Mitzenmacher, M., and Schafer, G.T., (2002), “The HuGS Platform: A Toolkit for Interactive Optimization,” *Advanced Visual Interfaces,* Trento, Italy.

Koza, J.R., (1992), “Genetic Programming: On the Programming of Computers by Means of Natural Selection,” *MIT Press,* Cambridge, MA.

Koza, J.R., (1994), “Genetic Programming II: Automatic Discovery of Reusable Programs,” *MIT Press,* Cambridge, MA.

Kraal, J.C., and Vance, J.M., (2001), “VEMECS: A Virtual Reality Interface for Spherical Mechanism,” *Journal of Engineering Design,* vol. 12 (3), pp. 245-254.

Kraemer, E., and Vetter, J., (1998), “Computational Steering,” *Proceedings of the Thirty-First Hawaii International Conference on System Sciences,* vol. 7, pp. 137-146.

Lamont, G. B., editor (1993), “Compendium of Parallel Programs of the Intel iPSC Computer,” *Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology,* Wright-Patterson AFB, OH 45433.

Lee, T., and Reitz, R.D., (2003), “Response Surface Method Optimization of a High-speed Direct-injection Diesel Engine Equipped with a Common Rail Injection System,” *Journal of Engineering for Gas Turbines and Power,* vol. 125 (2), pp. 541-546.

Liang, J.D., and Green, M., (1994), “JDCAD – A Highly Interactive 3D Modeling System,” *Computers & Graphics,* vol. 18 (4), pp. 499-506.

Longacre, K.D., Vance, J.M., and DeVries, R.I., (1996), “A Computer Tool to Facilitate Cross-attribute Optimization,” *Proceedings of the 6th AIAA MAO Symposium,* pp. 1275-1279.

Lotov, A.V., Bushenkov, V.A., and Kamenev, G.K., (2004), “Interactive Decision Maps,” *Kluwer Academic Publishers.*

Mahoney, P.D., (1995), “Driving VR,” *Computer Graphics World,* pp. 22-33.

McAllister, C.D., Simpson, T.W., and Yukish, M., (2000), “Goal Programming Applications in Multidisciplinary Design Optimization,” *8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization.*

McCorkle, D.S., K.M. Bryden, and C.G. Carmichael, (2003), "A new methodology for evolutionary optimization of energy systems," *Computer Methods in Applied Mechanics and Engineering*, Accepted for publication.

McCorkle, D.S., Bryden, K.M., and Kirstukas, S.J., (2003), "Building a Foundation for Power Plant Virtual Engineering," *Proceedings of the 28th International Technical Conference on Coal Utilization and Fuel Systems,* pp. 118-127.

Mena, J.B., and Malpica, J.A., (2003), "Color Image Segmentation Using the Dempster-Shafer Theory of Evidence for the Fusion of Texture," *ISPRS Archives,* vol. 34, Part 3/W8, Munich, Sept. 17-19.

Messac, A., (1996), "Physical Programming: Effective Optimization for Design," *AIAA Journal,* vol. 34 (1), pp. 149-158.

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

Michalewicz, Z., and Fogel, D.B., (2000), "How to Solve It: Modern Heuristics," *Springer*, Berlin.

Mistree, F., and Allen, J.K., (1997), "Position Paper Optimization In Decision-based Design," *DBD Workshop,* Orlando, Florida.

Okabe, A., Boots, B., Sugihara, K., and Chiu, S.N., (2000), "Spatial Tessellations – Concepts and Applications of Voronoi Diagrams," *Second Edition, Wiley Series in Probability and Statistics.*

Osborn, S.W., and Vance, J.M., (1995), "A Virtual Reality Environment for Synthesizing Spherical Four-bar Mechanisms," *Proceedings of the ASME Design Engineering Technical Conference,* vol. 2, pp. 885-892.

Pahl, G., and Beitz, W., (1984), "Engineering Design," *The Design Council*, London.

Papalambros, P.Y., (1994), "Model Reduction and Verification," *Advances in Design Optimization, Edited by Adeli, H., Chapman & Hall,* pp. 109-138.

Parker, S.G. and Johnson, C.R., (1995), "SCIRun: a scientific programming environment for computational steering," *Proceedings of Supercomputing'95.*

Parker, S., Weinstein, D., and Johnson, C.R., (1997), "The SCIRun Computational Steering Software System," *Modern Software Tools in Scientific Computing,* pp. 1-40.

Parmee, I.C., (1990), "Low-head Hydropower Systems," *PhD Thesis, Plymouth Polytechnic.*

Parmee, I.C., (2001), "Evolutionary and Adaptive Computation in Engineering Design," *Springer*.

Perles, B. and J.M. Vance, (2002), "Interactive virtual tools for manipulating NURBS surfaces in a virtual environment," *Journal of Mechanical Design*, vol. 124, pp. 158-163.

Rechenberg, I., (1973), "Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution," *Frommann-Holzboog Verlag*.

Reklaitis, G.V., Ravindran, A., and Ragsdell, K.M., (1983), "Engineering Optimization Methods and Applications," *John Wiley and Sons*.

Rittel, H., and Webber, M., (1973), "Dilemmas in a General Theory of Planning," *Policy Sciences,* vol. 4, pp. 155.

Ryken, M.J., and Vance, J.M., (2000), "Applying Virtual Reality Techniques to the Interactive Stress Analysis of a Tractor Lift Arm," *Finite Elements in Analysis and Design,* vol. 35 (2), pp. 141-155.

Salomon, D., (2002), "*A* Guide to Data Compression Methods," *Springer-Verlag,* New York.

Salomon, D., (2004), "Data Compression – The Complete Reference," *Springer*.

Schmitz, B., (1995), "Great expectations: the future of virtual design," *Computer-Aided Engineering*, vol. 14 (10), pp. 68-72.

Seth, A., Su, H., and Vance, J.M., (2006), "SHARP: A System for Haptic Assembly & Realistic Prototyping," *Proceedings of DETC'06 Computers and Information in Engineering Conference,* Philadelphia, PA.

Shine, W.B., and Eick, C.F., (1997), "Visualizing the Evolution of Genetic Algorithm Search Process,"

Shukla, R., Dragotti, P.L., Do, Minh., and Vetterli, M., (2002), "Improved Quadtree Algorithm Based on Joint Coding for Piecewise Smooth Image Compression," *Proceedings of IEEE International Conference on Multimedia and Expo,* Aug 26-29 2002, Lausanne, Switzerland.

Siddall, J.N., (1972), "Analytical Decision-Making in Engineering Design," *Prentice Hall,* New Jersey.

Siddall, J.N., (1979), *Transactions of ASME Journal of Mechanical Design,* vol. 101, pp. 674-681.

Siddall, J.N., (1982), "Optimal Engineering Design – Principles and Applications," *Marcel Dekker Inc*, New York.

Simon, H. A., (1960), "The New Science of Management Decision," *Harper and Row,* New York.

Simon, H.A., (1969), "The Sciences of the Artificial," *MIT Press,* Cambridge, MA.

Simon, H.A., (1984), "The structure of Ill-Structured Problems," in N. Cross (Ed.), *Development in Design Methodology*, John Wiley, Chichester.

Simpson, T.W., Mauery, T.M., Korte, J.J., and Mistree, F., (1998), "Comparison of Response Surface and Kringing Models for Multidisciplinary Design Optimization," *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization,* vol. 1, pp. 381-391.

Skarbek, W., and Koschan, A., (1994), "Color Image Segmentation," *Report, Technical Universit, Berlin.*

Suram, S., Ashlock, D.A., and Bryden, K.M., (2006), *"Graph Based Evolutionary Algorithms for Heat Exchanger Fin Shape Optimization,"* Accepted for Publication in the AIAA/MDAO.

Tan, K.C., Lee, T.H., Khoo, D., and Khor, E.F., (2001), "A Multi-objective Evolutionary Algorithm Toolbox for Computer-Aided Multi-objective Optimization," *IEEE Transactions on Systems, Man and Cybernatics-Part B: Cybernatics,* vol. 31 (4).

Tappeta, R.V., Renaud, J.E., Messac, A., and Sundarraj, G.J., (2000), "Interactive Physical Programming: Tradeoff Analysis and Decision Making in Multidisciplinary Optimization," *AIAA Journal,* vol. 38 (5), pp. 917-926.

Trigg, M.A., G.R. Tubby, and A.G. Sheard, (1999), "Automatic genetic optimization approach to two dimensional blade profile design for steam turbines," *Trans. ASME Turbomachinery*, vol. 121, pp. 11-17.

Urban, G.L., Bryden, K.M., and Ashlock, D.A., (2002), "Engineering Optimization of an Improved Plancha Stove," *Energy for Sustainable Development,* vol. 6 (2), pp. 5-15.

Vance, J.M., Larochelle, P., and Dorozhkin, D., (2002), "VRSPATIAL: Designing Spatial Mechanisms Using Virtual Reality," *Proceedings of DETC'02*.

Van Veldhuizen, D.A., and Lamont, G.B., (2000), "Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-art," *Evolutionary Computation,* vol. 8(2), pp. 125-148.

Vladimir, O.B., Christopher, C., Dipankar, G., Gary, Q., and Garret, N.V., (2002), "VisualDOC: A Software System for General-purpose Integration and Design Optimization," *AIAA/ISSMO 2002*.

Voland, G., (1999), "Engineering by Design," *Addison Wesley Longman*.

Walker, D.J., Dagger, B.K.J., and Roy, R, (1991), "Creative techniques in product and engineering design – A practical Workbook," *Woodhead Publishing Limited*.

Wang, L.H., Shen, W.M., Xie, H., Neelamkavil, J., and Pardasani, A., (2002), "Collaborative Conceptual Design – State-of-the-art and Future Trends," *Computer-Aided Design,* vol. 34 (13), pp. 981-996.

Wasfy, T.M., and Noor, A.K., (2001), "Visualization of CFD Results in Immersive Virtual Environments," *Advances in Engineering Software,* vol. 32 (9), pp. 717-730.

Wasfy, T.M., and Wasfy, A.M., (2003), "Strategy for Effective Visualization of CFD Datasets in Virtual Environments," *Proceedings of ASME Design Engineering Technical Conference on Computers and Information in Engineering*.

Wegman, E.J., and Symanzik, (2002), "Immersive Projection Technology for Visual Data Mining," *Journal of Computational and Graphical Statistics,* vol. 11 (1), pp. 163-188.

West, D.B., (1996), "Introduction to Graph Theory," *Prentice Hall,* Upper Saddle River, NJ.
Whitefield, R.I., Duffy, A.H.B., Coates, G., and Hills, W., (2000), "Coordination Approaches and Systems – Part I: A Strategic Perspective," *Research in Engineering Design,* vol. 12 (1), pp. 48-60.

Winer, E.H., and Bloebaum, C.L., (2001), "Visual Design Steering for Optimization Solution Improvement," *Structural Multidisciplinary Optimization,* vol. 22, pp. 219-229, Springer-Verlag.

Winer, E.H., and Bloebaum, C.L., (2002), "Development of Visual Design Steering as an Aid in Large-scale Multidisciplinary Design Optimization. Part I: Method Development," *Structural Multidisciplinary Optimization,* vol. 23, pp. 412-424.

Winer, E.H., and Bloebaum, C.L., (2002), "Development of Visual Design Steering as an Aid in Large-scale Multidisciplinary Design Optimization. Part II: Method Validation," *Structural Multidisciplinary Optimization,* vol. 23, pp. 426-436.

Woodson, T.T., (1966), "Introduction to Engineering Design," *McGraw-Hill,* New York.

Wu, A.S., De Jong, A., Burke, D.S., Grefenstette, J.J., and Ramsey, C.L., (1999), "A Visual Analysis of Evolutionary Algorithms," *Proceedings of the Congress on Evolutionary Computation, IEEE Press,* vol. 2, pp. 1419-1425.

Xiao, A., and Bryden, K.M., (2004), "Virtual Engineering: A Vision of the Next-Generation Product Realization Using Virtual Reality Technologies," *Proceedings of DETC/CIE 2004,ASME Design Automation Conference*.

Xiao, A., Bryden, K.M., Engelbrecht, J., Huang, G., and McCorkle, D.S., (2004), "Acceleration Methods in the Interactive Design of A Hydraulic Mixing Nozzle Using Virtual Engineering Tools," *Proceedings of ASME International Mechanical Engineering Congress*.

Xiao, A., Bryden, K.M., and McCorkle, D.S., (2005), "VE-Suite: A Software Framework For Design-Analysis Integration During Product Realization," *Proceedings of DETC/CIE 2005, 25th Computer and Information in Engineering*.

Yang, H., and Xue, D., (2003), "Recent Research on Developing Web-based Manufacturing Systems: A Review," *International Journal of Production Research,* vol. 41 (15), pp. 3601-3629.

Yeh, T.P., (1997), "Applying Virtual Reality Techniques to Engineering Design Optimization," *PhD Thesis, Iowa State University*.

Yeh, T.P., and Vance, J.M., (1998), "Applying Virtual Reality Techniques to Sensitivity-based Structural Shape Design," *Journal of Mechanical Design,* vol. 120 (4), pp. 612-619.

Yun, H., and Reitz, R.D., (2005), "Combustion Optimization in the Low-temperature Diesel Combustion Regime," *International Journal of Engine Research,* vol. 6 (5), pp. 513-524.