A Design Tool Architecture for the Rapid Evaluation of Product Design Tradeoffs in an Internet-based System Modeling Environment

by

Jacob Wronski

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2005
[June 2005]

Author .....
Department of Mechanical Engineering
May 1, 2005

Certified by .....
David R. Wallace
Esther and Harold E. Edgerton Associate Professor
Thesis Supervisor

Accepted by .....
Lallit Anand
Chairman, Department Committee on Graduate Students

A Design Tool Architecture for the Rapid Evaluation of Product Design Tradeoffs in an Internet-based System Modeling Environment

by

Jacob Wronski

Submitted to the Department of Mechanical Engineering
on May 1, 2005, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

This thesis presents a computer-aided design tool for the rapid evaluation of design tradeoffs in an integrated product modeling environment. The goal of this work is to provide product development organizations with better means of exploring product design spaces so as to identify promising design candidates early in the concept generation phase. Ultimately, such practices would streamline the product development process.

The proposed design tool is made up of two key components: an optimization engine, and the Distributed Object-based Modeling Environment. This modeling environment is part of an ongoing research initiative at the Computer-Aided Design Lab. The optimization engine consists of a multi-objective evolutionary algorithm developed at the Ecole Polytechnique Fédérale de Lausanne.

The first part of this thesis provides a comprehensive survey of all topics relevant to this work. Traditional product development is discussed along with some of the challenges inherent in this process. Integrated modeling tools are surveyed. Finally, a variety of optimization methods and algorithms are discussed, along with a review of commercially available optimization packages.

The second part discusses the developed design tool and the implications of this work on traditional product development. After a detailed description of the optimization algorithm, use of the design tool is illustrated with a trivial design example. Enabled by this work, a new "target-driven" design approach is introduced. In this approach, individuals select optimal tradeoffs between competing design objectives and use them, as design targets, to configure the integrated product model so as to achieve best-overall product performance. Validation of this design approach is done through the design of a hybrid PV-diesel energy system for two different applications. It is shown that the design tool effectively evaluates design tradeoffs and allows for the rapid derivation of optimal design alternatives.

Thesis Supervisor: David R. Wallace
Title: Esther and Harold E. Edgerton Associate Professor

Acknowledgments

First, I would like to thank The Center for Innovation in Product Development who, through their generosity, have made this research possible. Also, I would like to acknowledge the Ford Motor Company for providing context for this work.

I would especially like to thank my advisor, Professor David Wallace. First, for his generosity in welcoming me to his lab and for the wonderful opportunities this experience has provided me with. Second, for his approachability and willingness to help with ideas, which made significant contributions to this work. Thank you.

To all my lab-mates. I would like to express my sincerest gratitude to Elaine Yang for her help in implementing some of the ideas discussed in this work. Also, to: Amy Banzaert, Qing Cao, Twiggy Chan, Iason Chatzakis, Charles Dumont, Bill Fienup, Renu Fondeker, Sangmok Kim, Barry Kudrowitz, Wei Mao, Sittha Sukkasi, and Keith Thoresz. You were all a valuable part of my education at MIT.

To all my friends outside of my lab, for all the wonderful distractions you've provided me with. Especially, I would like to thank Joanna Gyory. For your kindness, encouragement ... and for making sure that I ate well while working on this paper. I am truly blessed and honoured to have your friendship.

Finally, I would like to thank my family. To my parents, Andrzej and Maria Wronski, none of this would have ever been possible, had it not been for your love, encouragment and support! I will forever be grateful for everything you've done for me. And to my brother and sister, for your great sense of humour, which has kept me sane through some of the more difficult times.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Product development firms seek to produce better performing products while reducing the time-to-market and costly design, build, test and refine cycles [77]. To achieve these objectives, various design tools and strategies are employed to accurately define the most promising concepts early in the design process. However, it has been observed that many computational tools are incapable of adapting to the dynamic and evolving nature of most product development environments and thus, the prediction of product performance early in the design process becomes a difficult task.

Organizations widely recognize the need for tools that facilitate the evaluation of integrated product performance. Thorough exploration of design alternatives early in the concept development phase reduces the likelihood that the development team will stumble on a superior design later on in the development process or that a competitor will introduce a similar product with dramatically better performance [72]. In practice, barriers such as system complexity and size, heterogeneous nature of subsystem models, and accessibility to models or expertise due to logistical and proprietary issues, greatly prohibit this approach [63]. Consequently, the evaluation of a single design candidate can take up to months to complete.

The multi-disciplinary nature of most product development teams presents a second challenge in product development, namely the management of design trade-offs in a way that maximizes the success of the product. For example, an airplane can be made lighter, but doing so will probably increase its manufacturing cost. Team members often approach such conflicts with a selfish attitude,

wishing to optimize the design objectives reflecting their own area of expertise. Clearly, this is not an acceptable approach and a better method of negotiating design tradeoffs, so as to improve the overall product performance, is needed.

To facilitate a better design approach, a design tool has been developed to rapidly evaluate and elucidate optimal design tradeoffs in an integrated product system model. This design tool is built into the Distributed Object-based Modeling Environment (DOME) and employs a heuristic optimization algorithm called QMOO [46]. Using the integration capabilities of DOME, the evaluation time of a single design candidate can be significantly reduced [77]. Furthermore, since the design tools employs an evolutionary algorithm to explore the design space of the product, the search for new product configurations is directed, where each new design candidate is "better" than its predecessors.

This thesis introduces a new design approach enabled by the design tool developed in this work. Members of a product development team can readily evaluate and obtain a graphical representation of the optimal tradeoffs of any integrated product model in the DOME framework. Furthermore, individuals can select a set of optimal design tradeoffs and use it to configure the integrated product model. The optimal design tradeoffs become the design targets for the design objectives of interest and the product system model seamlessly configures itself to satisfy these target values. "Target-driven" design and its validation in the field of product development is the main focus of this thesis.

1.2 About this Thesis

1.2.1 Chapter Two: Background

This chapter presents background topics relevant to the design tool and methodologies introduced in this work. First, the generic product development process is described and definitions of related terminology are provided. Throughout this discussion, the author presents a number of challenges, which product development organizations must overcome if they are to be successful. Next, a survey of existing integrated design tools is presented with emphasis on DOME. DOME is the design framework for which the optimization tool, presented in this thesis, is developed. It was chosen for its adaptability and flexible architecture, allowing design tools to be readily implemented in its platform. Subsequently, various optimization methods and algorithms are reviewed to provide

background for the optimization engine employed by the design tool in this work. Discussed optimization methods include: dynamic programming, linear optimization, calculus-based, enumerative and heuristic search techniques. Finally, existing design tools with multi-disciplinary optimization capabilities are discussed and compared to the tool developed in this work.

1.2.2 Chapter Three: The Queuing Multi-Objective Optimizer

Chapter 3 presents QMOO, a multi-objective, evolutionary algorithm developed by Leyland [46]. Initially, reasons for selecting QMOO as the optimization engine are presented. Next, the design features of QMOO are discussed. QMOO is a robust, steady-state, parallel, multi-objective optimizer, with an asynchronous architecture that makes it quite suitable for the computation environment of DOME. Following an overview of its design features, the algorithm itself is discussed in detail. Topics discussed include: assignment of parameter values, evaluation of objective functions, grouping, ranking, parent selection and thinning. Finally, design features of QMOO that allow it to run optimization problems in parallel are reviewed.

1.2.3 Chapter Four: Target-Driven Design

This chapter introduces a more effective, target-driven, approach to product design. First, a trivial design example of a tube-bundle heat exchanger is presented. Next, the optimization tool developed in this work is discussed in detail. Using the design example, the author illustrates how a designer may use this tool to build an optimization model, publish it on a live Internet server, run its underlying simulation and evaluate its results. Next, the communication interface between the optimization engine, QMOO, and DOME is discussed. The DOME evaluator data object is responsible for sending parameter values from QMOO to an integration model in DOME, and receiving objective function values from the integration model so that they may be assigned to individuals and managed by QMOO. The DOME monitor is responsible for passing information about the optimization solution to the designer running the simulation. Finally, implications of this work on the traditional product development process are discussed. In short, the optimization tool enables product development teams to effectively run a high number of "what-if" scenarios on a product system model. Furthermore, it rapidly evaluates and elucidates design tradeoffs, allowing the team to negotiate them in a way that is optimal with respect to the overall performance of the product.

1.2.4 Chapter Five: Application

The optimization tool has been applied to an engineering problem to validate target-driven design. This problem has been developed and studied extensively by other researchers at the CADLab. In this design scenario, the optimization tool was applied successfully and proved to be a valuable tool when assessing the performance tradeoffs between different design configurations.

The problem was originally proposed by Sukkasi [70] as part of the *Alternative Energy Design Toolkit* and deals with the design of a hybrid PV-diesel energy system. Such energy systems offer an alternative source of electricity for remote locations far from established utility grids. The PV array in the system can deliver clean electricity, but at a high capital cost. Alternatively, the diesel engine delivers inexpensive electricity, but it also taxing on the environment. The optimization consists of finding a number of design configurations, which produce electricity at minimum cost and impact to the environment, and then evaluating the tradeoffs between different designs.

Chapter 2

Background

2.1 Introduction

To properly introduce the design analysis tool developed in this work, and its implications on the traditional product development process, several relevant topics must first be discussed. Traditional product development process and its inherent challenges will be presented. Integrated design tools, including DOME, intended for better management and streamlining of the product development process, will be explored. The discussion will then review the more theoretical topic of optimization algorithms, to provide relevant background to the design analysis tool presented in later chapters. Finally, multidisciplinary optimization tools, similar to the one developed in this work, will be reviewed.

2.2 Product Development

This section will provide an in-depth understanding of the product development process. First, a number of definitions, used throughout this work, will be presented. Next, a generic product development process will be introduced. Finally, challenges encountered by design teams during product development will be presented.

Chapter 4 will present a design tool, developed in this work, which empowers product development teams to effectively overcome common problems (see §2.2.3) encountered in product development projects.

2.2.1 Terminology

A *product* in this work refers to any engineered, discrete and physical object that can be sold by an enterprise to its customers. Examples of products include kitchen appliances, sail boats, or portable computers. *Product development* (or *design*) is the sequence of steps or activities, which an enterprise employs to conceive, design, and commercialize a product [72]. Products are often developed by multidisciplinary teams with unique areas of expertise reflecting the nature of the product. They represent *product development organizations* and the present discussion will focus on product development in this context.

2.2.2 The Product Development Process

Product development is the process of transforming a particular costumer need to a final product or service that addresses that need [4]. It is an interdisciplinary activity requiring contributions from all functions of an organization including design, manufacturing, and product marketing. Product development is commonly viewed as an iterative design-build-test cycle driven by monetary and time constraints imposed by project management on the development team. Many generic variants of this process have been presented in literature: Ulrich and Eppinger [72], Hyman [38], Eggert [24], Cagan and Vogel [9]. Figure 2-1 illustrates one example, namely the spiral product development process, reproduced from Ulrich and Eppinger [72].

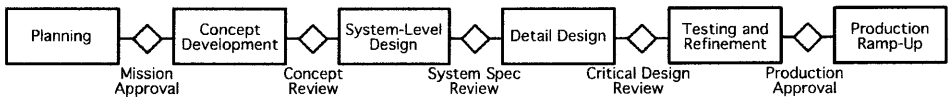


Figure 2-1: Process flow diagram for a generic product development process. Ulrich and Eppinger [72] (p. 23).

Product development begins with project planning, an activity often referred to as phase zero since it precedes project approval and launch of the actual product development process [72]. This phase begins with an assessment of current technology, social trends, and economic trends for the purpose of identifying product opportunities. The output of the planning phase is a mission statement, which includes business goals, target-market for the product and anticipated challenges that will guide the development effort.

The next phase, concept development, begins with the identification of customer needs. This task is not trivial and a product developed without a good understanding of the customer will not succeed. Customers often express their preferences by describing a solution concept or an implementation approach for example, a customer might say, "I would like it to be electrically operated". A designer must be able to filter through such user input and obtain a need statement that is expressed in terms, independent of a particular technological solution [72]. One example of a successful product addressing a set of user needs without any preconceived hypothesis about the technology to address that need is the *iMac*™computer developed by Apple™[9]. Designers and engineers understood that consumers were tired of the simple and dull gray desktop computers on the market. Making the *iMac*™computer transparent and translucent and introducing candy colours was an instant success. For the first 139 days from its debut, an iMacł sold at the rate of four a minute, every hour, of every day[1].

Customer needs are generally expressed in the "language of the customer" [72]. While they do provide knowledge about issues of interest to the customer, they provide little guidance about how to design the actual product. Designers must effectively translate the customer needs into product target specifications a precise description of what the product has to do. Target specifications are used during the concept development phase to evaluate the relative quality of generated concepts.

The concept generation phase can be one of the most creative and rewarding activities for a product development team. It involves a mix of creative problem solving, benchmarking of competitive products, consulting with external experts and the systematic evaluation of various concepts through physical prototypes or analytical models. Typically, a development team will generate hundreds of concepts of which one will be selected for detailed design. The result of the concept development phase is a revised and detailed list of final specifications of what the product must do.

Once a design concept has been selected and final product specifications are in place, detailed design takes place. At the system level, product architecture is defined and the product is fragmented into its subsystems and components. The geometry, materials, and tolerances of all components in the product are specified. Process plans for fabrication and assembly of the product are devel-

[1]"Yum./Apple iMac", *Innovation*, Fall, 1999, p.76.

oped. CAD tools are used frequently to describe product geometry. Design teams often approach this phase with a "design for X" (DFX) methodology, where X may correspond to one of many criteria, such as manufacturability, environment or robustness. This often leads to competing design objectives. For example, designers often consider the cost of manufacturing a product versus its environmental impact. Such design tradeoffs are not trivial to resolve.

The testing and refinement phase involves the construction and evaluation of multiple pre-production versions of the product. *Alpha* and *Beta* prototypes of the final design are often employed at this stage to answer questions about performance and reliability.

Production ramp-up is the final phase of the product development process. At this stage, the product is entirely fabricated using the intended production system, work force is trained, and any remaining problems in the production process are quickly resolved. The product development process ends when the product is launched to the consumer market.

For a more comprehensive discussion of product development, the reader is referred to Ulrich and Eppinger [72].

2.2.3 Challenges Facing Product Development Organizations

Development of successful products is a difficult task. The size and scope of most product development projects creates a number of challenges for collaborative engineering in a multidisciplinary environment. This section will focus on two specific ones: (i) creating an integrated product model, and (ii) resolving design tradeoffs during product development.

The need to develop integrated models to predict the performance of a product during development is widely recognized. Computer simulations would allow design teams to run a high number of relatively inexpensive, "what-if" scenarios to determine the final product configuration. Thorough exploration of design alternatives early in the concept development phase will reduce the likelihood that the team will stumble upon a superior concept later in the development process or that a competitor will introduce a product with dramatically better performance [72]. In practice, comprehensive product system modeling has been deemed infeasible [72]. Cutkosky et al. [20] identified a number of specific issues: (i) the size, complexity, and evolving nature of a product prohibits a com-

plete product model definition, (ii) members of a design team often work with different tools, data management systems and representations, and (iii) logistical and proprietary issues make global data unavailable. As a result, a single design scenario takes weeks, sometimes months, to evaluate [77]. Product development teams often employ design-of-experiments (DOE) techniques [49, 30], which can minimize the number of experiments required to explore the design space of the product.[2] While this approach is effective, tools that allow development teams to evaluate the product system performance remain highly attractive. Cagan and Vogel [9] summarize this point succinctly:

> If used correctly, [integrated product development] will significantly reduce downstream development problems in parts integration, manufacturing quality, and missed opportunities in style and features of the product.

Rigid product integration defines the interactions between participants and tools globally in a top-down manner. Such an approach has a significant downside, as it slows the organization's ability to change rapidly and be innovative, because the cost and complexity of redefining the global interaction model is prohibitive [77]. A variety of integrated design tools will be discussed in §2.3.

Product development organizations often encounter design tradeoffs – inverse relationships between two specifications that are inherent in the selected product concept. For example, a design team working on a bicycle frame charged with the task of decreasing its mass is able to accomplish this task, but only with a more expensive material, such as aluminum instead of steel. Team members often approach such conflicts with a biased view, wishing to optimize design objectives that reflect their own area of expertise. Resolving such tradeoffs in a multidisciplinary development project is the most difficult part [72].

Existing integrated product modeling environments will be presented next. Chapter 4 will present a design tool developed for the DOME simulation environment, which facilitates the rapid elucidation and evaluation of optimal design tradeoffs, thus addressing the design challenges presented in this discussion.

[2] DOE methods stem from robust design theory. For a more detailed discussion on this subject, the reader is referred to Phadke [59].

2.3 Design Tools For Integrated Product Modeling

Integrated product development environments offer a streamlined development process, accelerating the time to market and improving product quality [63]. These driving forces are encouraging research teams and commercial enterprises to develop engineering tools that can facilitate this development approach. This section will provide an overview of existing environments that allow for design collaboration and knowledge sharing in a multidisciplinary product development organization. Greater detail will be afforded to the Distributed Object-based Modeling Environment (DOME), since the design tool introduced in this work was developed for this environment.

2.3.1 Collaboration-Oriented Design Tools

The SHARE project [71] is aimed to support design engineers or teams by allowing them to gather, organize, re-access and communicate design information over computer networks to establish a "shared understanding" of the design and development process. NEXT-LINK [58] incorporate agents to coordinate design decisions affected by specifications and constraints. Case and Lu [12] developed a software tool to support collaborative engineering design by treating interactions as a process of discourse. The model captures design commitments as opinions subject to review by other designers. Agents are also used to identify conflicts between designers and to negotiate their resolution. The *Electronic Design Notebook*, developed by Lewis and Singh [45], is an interactive electronic document, which maintains the look and feel of an engineering document to provide an integrated user interface for computer programs, design studies, planning documents, and databases. Bliznakov et al. [5] propose a design information system, which allows designers in a large virtual organization to indicate the status of tasks assigned to each designer or team so that other designers can follow their progress. This distributed information system is managed by a central database. Hardwick and Spooner [34] propose an information infrastructure architecture that enhances collaboration between design and manufacturing firms. This collaboration tool uses the WWW for information sharing and the STEP standard [52] for product modeling. N-dim is a computer-based collaborative design environment for capturing, organizing and sharing data [78]. It allows participants to define information types that capture the relation between data or models. A computer-based design system DICE, developed by Sriram and Logcher [69], provides a shared workspace where multiple designers work in separate engineering disciplines. DICE (Distributed and Integrated Environment for Computer-aided Engineering) implements an object-

oriented databases management system with a global control mechanism to resolve coordination and communication problems in product development projects. The design rationale provided during the product development process is also used for resolving design conflicts [57, 56]. Xiaojuan et al. [79] are developing iShare, an Internet-based middle-ware and collaboration technology with applications in integrated modeling.

A number of design tools, which facilitate integrated product development, are available commercially. FIPER®(The Federated Intelligent Product Environment), developed by Engineous Software™, provides organizations with a design infrastructure for process integration and enterprise-wide product optimization [3]. FIPER®implements a central control system, which manages workflow, job dispatching and execution, and interactions with third-party applications. Although the models may be distributed, they are not federated. ModelCenter®is a system integration package developed by Phoenix Integration™.[4] It implements a central "scheduler", which determines the model execution process based on a user pre-defined workflow. AML®, developed by TechnoSoft™, is an object-oriented, knowledge-based engineering modeling framework. AML implements a demand-driven system solving mechanism. Given a new model state, only those models that are dependent on the change will be re-solved so that a new, consistent, system state may be obtained. This ensures the efficient use of computational resources. CO®, developed by Oculus Technologies™, is an ad-hoc, peer-to-peer system integration tool[5]. CO integrates creates dynamic links between third party applications and propagates parameter value changes across those links. CO does not support any type of solving mechanism.

The different systems discussed here employ a wide variety of architectures, data models and communications technologies, but all rely on some form of a consolidated, explicit description of the complete integrated system model. This approach may work well for small systems, but becomes a formidable task for large-scale product development projects. The following section will discuss an integrated product-modeling environment that employs an emergent system integration approach, inspired by the World Wide Web.

[3] www.engineous.com
[4] www.phoenix-int.com
[5] www.oculustech.com

2.3.2 DOME (Distributed Object-based Modeling Environment)

The DOME integrated modeling environment was developed at the MIT Computer-Aided Design Laboratory (CADLab) to facilitate the creation of design models that can rapidly predict integrated product performance [53, 54]. The DOME framework asserts that multidisciplinary problems can be decomposed into modular sub-problems [63]. Modularity distributes knowledge and responsibility among members of a product development team [72]. Individuals create design models based on their area of expertise, and make them accessible over the Internet. Individuals can also define relationships between their simulation services and the services of other simulations available on the Internet. The resultant service exchange network becomes an emergent, distributed product system model capable of predicting the integrated behaviour of a design alternative [63].

Simulation services are models that represent elements of a complete product system. The underlying simulation can be a third-party application, such as Excel®, Matlab®, CATIA®, or a DOME-native mathematical model. Participants that build a model (see Figure 2-2), define parametric inputs, which drive the simulation, and parametric outputs, which are the simulation results. Input-output relationships, and other implementation details about the simulation are hidden from the subscribing user, respecting proprietary issues in a product development project.

Model owners are able to define and publish parametrically operable interfaces to their simulation on the Internet. Outside users are then able to interact with the underlying simulation through this interface. Figure 2-3 illustrates the procedure that an outside user would follow to interact with the volume service simulation described above.

To build integrated simulations, participants define relationships between design models in a declarative fashion, which does not require knowledge about the global view of the system integration or the sequence in which the relationships should be executed. Each simulation generates and provides a causal map in its interface that relates inputs to outputs. This causal map has enough information to understand parameter changes caused by external sources, and is able to coordinate solving of the interface in an efficient manner. This distributed solving architecture can be referred to as federated – models inter-operate at a global level, but maintain control at a local level. This ensures that computational efficiency is not sacrificed and the ability to detect degenerate causal structures is not lost in the emergent simulation environment. Most of all, this approach does not require the definition of a central workflow model to coordinate the solving of all relationships in the integrated

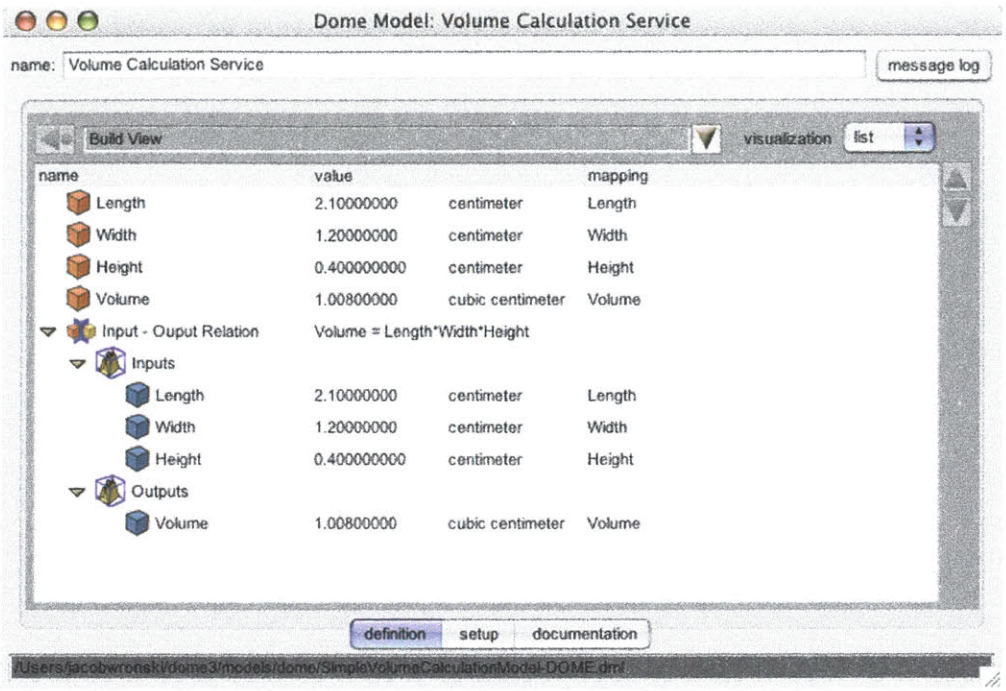


Figure 2-2: DOME-native model. This trivial service simulation calculates part volume.

model. The integrated system is able to emerge and evolve rapidly [76].

2.4 Optimization Methods and Algorithms

Numerical optimization techniques, when carefully applied, are an effective tool in engineering design. Many approaches to optimization exist, and can be grouped into five, broad categories: (i) dynamic programming, (ii) linear, (iii) non-linear or calculus-based, (iv) enumerative, and (v) heuristic[6] [31]. A comparative study will reveal that each method is unique, and suitable for a particular set of design problems. This section will review each optimization technique with focus on its application to real-world engineering problems. However, before proceeding with this discussion, a formal definition of an *optimization problem* will be presented.

2.4.1 Definition of an Optimization Problem

A definition of a *single-objective optimization problem (SOP)* is provided by Vanderplaats [74].

[6] *Goldberg* identified the fifth optimization method as random. We will however focus on heuristic search techniques, a subset of random search techniques.

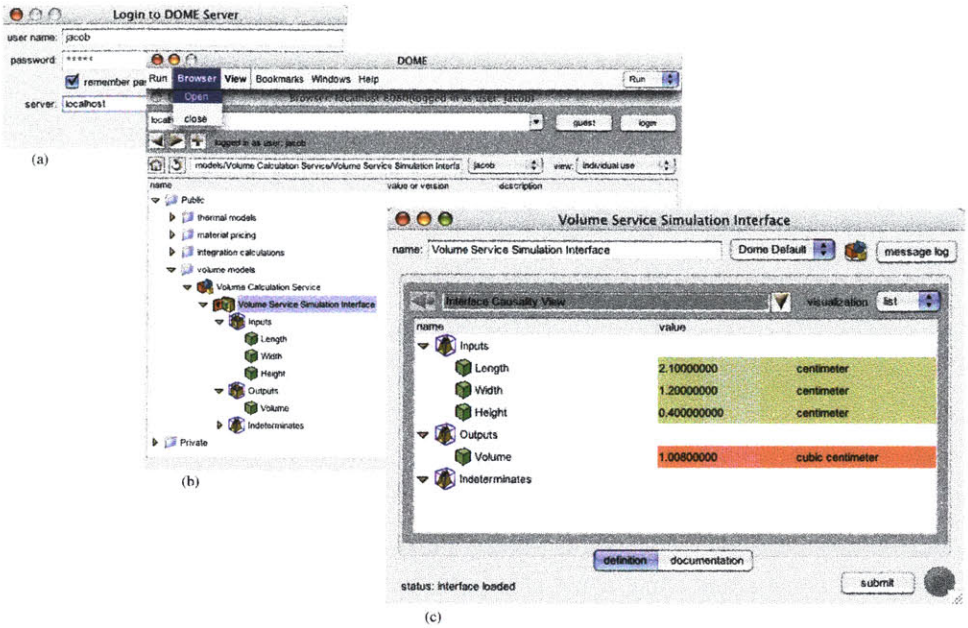


Figure 2-3: (a) Outside participant is logging into a DOME server on a remote machine. (b) Public folder space on the server, outside participant is opening the volume calculation service. (c) A DOME interface that the design participant must use to interact with the underlying simulation.

In mathematical terms, a SOP minimizes the objective function $f(x)$ where x is an n-dimensional vector $x = (x_1, \ldots, x_n)$ from some design space S.

Or, in general,

$$\text{minimize:} \quad f(\boldsymbol{x}) \tag{2.1}$$

$$\text{subject to:} \quad g_i(\boldsymbol{x}) \leq 0, \qquad i = 1, \ldots, l, \quad \boldsymbol{x} \in \mathcal{S} \tag{2.2}$$

$$h_k(\boldsymbol{x}) = 0, \qquad j = 1, \ldots, m \tag{2.3}$$

$$x_i^l \leq x_i \leq x_i^u, \quad k = 1, \ldots, n \tag{2.4}$$

Equations (2.2) and (2.3) define the inequality and equality constraints, respectively. Equation (2.4) defines the lower and upper limits of the search space for each decision variable.

Therefore, given a design vector with n decision variables, one would like to find a set of values, within the specified range, such that they yield the minimum possible value of $f(x)$, while satisfying all constraints imposed on the problem.

Van Veldhuizen and Lamont [73] provide a similar definition for a *multi-objective optimization problem (MOP)* – the difference being that the goal is to minimize, not one objective function, but an n-dimensional vector of objectives $F(x)$. In a MOP, the desired result is a set of solutions, which illustrate the optimal tradeoffs between design objectives.

2.4.2 Dynamic Programming Techniques

This method is most useful in design problems configured in stages, and whose design can be characterized as a sequence of design decisions made in each stage. The technique can be applied to any optimum design problem for a system with n stages that requires a design decision at each stage i. The design decision at each stage transforms the condition, or state, of the system entering stage i into an outgoing state that serves as the entering state for the next stage $i + 1$.

The first step in formulating a dynamic programming problem is establishing a design objective (the system state) to be optimized. Although generally not applicable to multi-objective design problems, weighting could be applied to transform a multi-objective problem into a single-objective problem based on some measure of importance. Decision variables must also be selected. Finally, constraints are applied to the decision variables and the optimization problem is fully defined.

The optimization analysis involves stepping forward (or backward) through the system, and at each stage in the system calculating the design objective for all available decision variables. Design configuration that results in the best performance, as measured by the objective function, is selected at each stage and becomes the input for the subsequent stage. This process is repeated until the final stage of the design system is reached.

Dynamic programming is a useful engineering tool in operations research the study of optimal resource allocation. One example of its application is the planning of transmission line routes in the power industry, where one is often concerned with minimizing the cost of transmission lines.

2.4.3 Linear Optimization Techniques

This widely studied class of optimization problems involves inequality constraints, which consist of both an objective function U and a set of constraints that are linear functions of the design variables. Mathematical formulation of the linear programming problem is taken from Hyman [38]:

> Minimize (or maximize) the linear objective function $U(x)$ where x is an n-dimensional decisions variable vector $x=(x_1,\ldots,x_n)$ from some universe S, subject to m constraints.
>
> Or, in general,

$$\text{minimize:} \quad U(\boldsymbol{x}) = \sum_{i=1}^{n} k_i x_i \tag{2.5}$$

$$\text{subject to:} \quad \sum_{i=1}^{n} a_{ij} x_i \leq r_j, \qquad j = 1, \ldots, m \tag{2.6}$$

$$x_i \geq 0 \tag{2.7}$$

> In the above expression, a_{ij} and k_i are constants presumed to have known values in any particular problem. Also, the standard form of linear programming problems requires that the design variables be non-negative, as noted by equation (2.7).

In linear programming problems, the optimal solutions occur at the "vertices" of a design space[7]. Therefore, instead of looking at an infinite number of possible solutions, one only has to evaluate the objective function at the corner points of the design space to discover the optimum. This has great significance on the development of an efficient linear programming algorithm.

The *Simplex* algorithm is a popular linear programming approach. To begin the analysis, an appropriate corner point is selected and the objective function is evaluated at that point. Next, the algorithm moves along a pre-determined edge of the polygon until a vertex is reached, and the objective function is evaluated at that point. The most efficient strategy is to move along an edge for which the objective function increases most rapidly. The algorithm continues to move from vertex to vertex until a maximum (or minimum) for the objective function is found.

Linear programming is limited to single-objective, linear engineering problems, for which the design objective and constraints can be written in mathematical form. The optimal allocation of man-

[7]For a formal proof, the reader is referred to Gottfried and Weisman [33]

ufacturing resources in a plant is one application. However, most engineering problems of interest are not linear, and even fewer can be formulated in mathematical terms. Algorithms, such as the *Sequential Linear Programming (SLP)* technique developed by Kelley [40], use Taylor-series expansion to approximate non-linear problems as piece-wise linear. This approach requires an advanced understanding of the design space of the engineering problem at hand, and should only be used with caution. For a more detailed description of linear programming algorithms, the reader is referred to [15].

2.4.4 Non-Linear or Calculus-Based Techniques

Calculus-based techniques use derivatives of the objective function, to obtain the optimal system state[8]. They can be subdivided into two main classes: (i) direct methods, and (ii) indirect methods [31].

Direct methods are applied to unconstrained optimization problems. They seek local optima by hopping on the objective function and moving in the direction related to the local gradient. This is simply the notion of *hill climbing* – to find the local best, climb the function in the steepest direction. Calculus dictates that one can expect to find the minimum or maximum of a function at a point where the gradient of the function is null. The earliest gradient algorithm known is the *steepest descent method* developed by Cauchy [13] in 1847. This simple algorithm is written as follows:

Data: $x_o \in \Re^n$

Step 0: set $i = 0$

Step 1: compute the search direction

$$h_i = -\nabla \boldsymbol{f}(x_i)$$

stop if $\nabla \boldsymbol{f}(x_i) = 0$

Step 2: compute the step size λ

Step 3: set

$$x_{i+1} = x_i + \lambda h_i$$

replace i by $i+1$ and go to Step 1

[8]Many practical parameter spaces have little respect for the notion of a derivative and the smoothness this implies – immediately the shortcomings of this method present themselves.

Determining the appropriate step size is perhaps the most difficult task. A small step size will result in a laborious method in reaching the function minimum, while a large step size will result in the problem minimum being missed completely. For this reason, the steepest descent method will seldom converge to a design space minimum. Newtons method [51] uses first and second derivatives to construct a quadratic approximation of an objective function at a specific point. This approximated function is then minimized, and the decision variables at the minimum value are used as the starting point for the next iteration. This method performs better than the steepest gradient method if the initial point is chosen close to the optimal value. The conjugate gradient method, first proposed by Hestenes [35] in 1952, is an improvement of the steepest descent method. This method uses a combination of the local gradient, and the gradient of the previous iteration, to determine the new direction of motion. Other, more powerful, calculus-based methods for unconstrained optimization are known as Quasi-Newton Methods. The DFP algorithm, developed by Davidon [21] and Fletcher and Powell [28], and the BFGS algorithm, developed independently by Broyden [7], Fletcher [27], Goldfarb [32] and Shanno [64] are both quasi-Newton methods. Chong and Zak [15], provide a detailed review of both algorithms.

Indirect methods are suitable for constrained optimization problems. They seek local optima by solving the usually nonlinear set of equations resulting from setting the gradient of the objective function equal to zero [31]. In constrained optimization problems, an optimum design variable vector minimizes the objective function while satisfying all design constraints. *Penalty and barrier methods* treat the optimization problem as unconstrained, and assign penalties to the objective function to limit constraint violations. Carrol and Johnson [11] introduced an interesting penalty function method, which has been developed extensively by Fiacco and McCormick [26] into the *Sequential Unconstrained Minimization Technique (SUMT). Sequential Quadratic Programming (SQP)* [55] was developed in the 1970s and is considered to be the most efficient gradient-based optimization technique. The method involves creating a quadratic approximation to the Lagrange-Newton formulation of the optimization problem and a linear approximation to the problem constraints. These equations are then solved using quadratic programming techniques until some termination criteria are reached. *Method of Feasible Directions* is a calculus-based optimization technique proposed Vanderplaats [74] in 1984. The optimization begins with a point in the feasible design region (no active or violated constraints), and moves in the direction of steepest descent. Conjugate direction or variable metric techniques are used on the objective function until a constraint is encountered.

At the constraint boundary, the algorithm must find a usable-feasible direction, where a usable direction is one that reduces the objective and a feasible direction is one that either follows or moves inside a constraint boundary. This requires that a system of equations be solved numerically until some termination criteria is reached. Numerous other constrained optimization algorithms exist, including the *Augment Lagrange Multiplier (ALM)*, the *Genralized Reduced Gradient (GRG)* and the *Mixed Integer Programming* methods, as examples. Vanderplaats [74] describes these techniques in detail.

Calculus-based optimization techniques have two inherent shortcomings. First, they are local in scope – the optima they seek are the best in a neighbourhood of the current point. For example, in a multimodal design space, using a gradient-based method and starting the search closer to a lower peak will cause the algorithm to completely miss the higher peak. Second, calculus-based methods depend on the existence of derivatives. Practical design spaces are fraught with discontinuities and vast multimodal, noisy search spaces. As a result, calculus-based techniques are insufficiently robust and require a high level of understanding of the design problem to which they are applied. Use of such techniques should be left to the most experienced designers.

2.4.5 Enumerative Techniques

Enumerative optimization techniques are fairly straightforward. Given a finite search space, or a discretized infinite search space, the search algorithm evaluates the objective function at every point in the space, one at a time. Although this type of "trial-and-error" search is common in engineering practice, many practical design spaces are simply too large making this method inefficient. However, given infinite time, the enumerative search technique is guaranteed to locate the optimum points in the types of design spaces discussed above.

2.4.6 Heuristic Techniques

Heuristic optimization methods employ "self-educating" techniques to effectively explore the parameter design space. They are self-educating because with subsequent iterations, they select design vector configurations, which result in the improved performance of the objective function. In the long run, they can be expected to do no better than enumerative schemes, but they far more efficient and practical to implement. However, in practice the best configuration of a given problem instance is seldom required. More realistically, an acceptable configuration should be produced with rea-

sonably limited computer resources. This section will present two of the most popular heuristic search techniques: (i) simulated annealing, and (ii) evolutionary algorithms. The evolutionary algorithms technique can be implemented with population-based algorithms. Unlike gradient-based search techniques, which present a single optimum design point, population-based algorithms can present the decision maker with a set of optimal designs, which illustrate the interesting regions of the design model.

Simulated Annealing

Krikpatrick et al. [44] introduced the concept of optimization by simulated annealing in 1983. Derived from statistical mechanics, it is a heuristic technique that mathematically mirrors the cooling of a material to a state of minimum energy. Jilla and de Weck [39] use the following analogy to describe this process:

> If a liquid material (i.e. metal) cools and anneals too quickly, then the material will solidify into a sub-optimal configuration. If the liquid material cools slowly, the crystals within the material will solidify optimally into a state of minimum energy (i.e. ground state). This ground state corresponds to the minimum of the cost function in an optimization problem.

Returning to optimization, simulated annealing is the process of varying a set of independent design parameters to minimize the energy state (or value) of a given objective function, subject to some annealing schedule (or termination criterion). Krikpatrick et al. [44] discuss four key elements that a formulated problem must exhibit in order to be suitable for optimization using simulated annealing. First, one must define a vector of independent parameters, which will represent the state (or configuration) of the design inside the parameter space. Second, a random generator must be implemented to explore the neighbourhoods around existing individuals. New design points are generated by perturbing existing individuals in one or more of the given design parameters – also called degrees of freedom. The number of design parameters allowed to undergo random perturbations must be carefully considered. If the entire design vector is allowed to change, the simulated annealing search degenerates to a random search. The other extreme – only one parameter is allowed to change, will result in constrictive neighbourhoods, lack of diversity and limited exploration of the decision variable space. The third element that must be defined is a quantitative objective function. If the

problem of interest requires the optimization of multiple design objectives, weighting factors can be used to aggregate all of the objectives into a single objective function. The fourth element to be defined is an annealing schedule, or termination criterion. A common cooling schedule is to reduce the system temperature by a constant fraction with every generation. For example, if the system begins with a temperature of 1000, and continues at a reduction rate of 95% with each perturbation, the algorithm will run for approximately 100 generations. The system temperature also plays an important role in the preservation of diversity during the optimization process. This will be discussed shortly.

The simulated annealing algorithm is presented below:

Step 1: choose a random individual (Γ_i), select the initial system temperature and outline the cooling (i.e. annealing) schedule

Step 2: evaluate the system energy (i.e. objective function) of the optimization model $\boldsymbol{E}(\Gamma_i)$

Step 3: perturb the current individual in one or more degrees of freedom to obtain a neighbouring individual (Γ_{i+1})

Step 4: evaluate the system energy of the new individual $\boldsymbol{E}(\Gamma_{i+1})$

Step 5: if $\boldsymbol{E}(\Gamma_{i+1}) < \boldsymbol{E}(\Gamma_i)$, Γ_{i+1} is the new current solution

Step 6: if $\boldsymbol{E}(\Gamma_{i+1}) > \boldsymbol{E}(\Gamma_i)$, then accept Γ_{i+1} as the new current solution with a probability $e^{(-\Delta/T)}$ where $\Delta = \boldsymbol{E}(\Gamma_{i+1}) - \boldsymbol{E}(\Gamma_i)$

Step 7: reduce the system temperature according to the cooling schedule

Step 8: terminate the algorithm when the termination criterion has been reached

The reason for accepting an inferior solution (step 6) is to prevent the algorithm from getting trapped in a local optimum. The probability of accepting an inferior solution is dependent on two parameters: (i) the system energy difference (Δ) between two neighbours, and (ii) the system temperature. Therefore, the probability of moving to a worse solution with the hope of escaping a local minimum decreases over time as the system cools. Finally, there are many items one may tailor within the algorithm to affect its performance, including initial system temperature, cooling schedule, and DOF within a neighbourhood.

Evolutionary Algorithms

Evolutionary algorithms (EAs) are optimization techniques inspired by organic evolution observed amongst living organisms. In evolution, genetic variability from generation to generation significantly affects the probability that a given individual will survive and produce offspring. Natural selection, the mechanism of evolution, exerts survival pressure on the population and ensures that only individuals with favourable genotypes are able to reproduce successfully [60]. This process is commonly referred to as survival of the fittest. Likewise, in an EA, a *population* of *individuals* is *evolved* toward the solution of an optimization problem by undergoing a number of *operations*, which produce new individuals and remove existing ones [46].

An *individual* is a possible solution to an optimization problem. It contains a design vector $\boldsymbol{x}$, which represents where the individual lies in the decision variable space, and also information about its degree of optimality, usually assessed via the design objective function.

The population evolves towards better solutions by creating progeny superior in fitness to their parents, and removing individuals from the population found to be lacking.

New individuals are added to the population by *crossover* operations. Crossover involves selecting two (or more) individuals from a population and creating a *child* (or children) that in some way resembles its *parents*, much like sexual reproduction. A good implementation of an EA will use a selection scheme with bias towards better parts of the population, much like a breeder of roses will choose individuals best fitting his or her goals, to be parents for the next generation. *Mutation* is an important operator, which takes an individual and modifies some of its parameters, much like the natural mutation of the genetic code of an organism. In an EA, the role of mutation is to ensure preservation of diversity – the spread of the population in the search space. Mutation operators, however, must be used cautiously. Too little mutation will lead to loss of diversity in the population resulting in the algorithm converging quickly to a narrow region of the design space, and thus getting trapped in some local optima. Too much mutation will decrease the population convergence rate, thus causing the population to explore a wide region of the design space at the expense of computational time.

Removal of individuals from the population is dependent on the structure of the EA. In *generational* EAs, the population is processed a generation at a time, where a number of children equal to the size

of the population is generated, and used to entirely replace the existing population. In *elitist* EAs, a few of the fittest individuals in the population are retained from generation to generation, acting as a marker of the best performance of the algorithm thus far. These individuals also contribute to the overall convergence of the algorithm by having more children. In *steady-state* EAs, individuals are added and removed from the population as necessary and very often the criteria for removal of points results in some measure of elitism.

The basic overview of EAs thus complete, the discussion will now focus on each step of the algorithm in greater detail. In its most general form, an EA will follow the following steps:

Step 1: define a decision variable encoding – decoding scheme

Step 2: define a "fitness" function $\mathcal{F}$

Step 3: initialize a *population* of *individuals* in some search space $\mathcal{S}$

Step 4: generate a number of new *individuals* and add them to the population

Step 5: remove a number of *individuals* from the *population*, which are inferior in terms of the fitness function $\mathcal{F}$

Step 6: continue until some termination criterion is reached

Encoding is used extensively in one family of EAs, namely genetic algorithms. It involves 'coding' all of the individual's decision variables into a different alphabetical representation, usually binary. The crossover and mutation operators then work directly on this new representation. Thus a mutation operator would choose whether or not to change a particular bit, and a crossover operator would choose individual bits from each parent. Decoding is the reverse process of encoding, where decision variables are translated back to their natural form. Binary encoding of decision variables presents a number of issues. First, the number of bits used for the representation of a real variable must be carefully chosen so as to ensure sufficient resolution of the problem's optima. Secondly, representation of integers is difficult in problems where the feasible range is not a power of 2, and requires that the integers be over specified (i.e. using more *bits* than is necessary). Another drawback of encoding variables to bits is apparent in the mutation operator. Since mutation works at the bit level, a good parent with a value of 1 (a binary encoding of 000001) is as likely to produce a child with a value of 33 (100001) as 0 (000000), although common-sense assumptions about the location of optima would suggest that 0 is a stronger individual than 33. A further problem is that it is extremely difficult to get from a value of 31 (011111) to a value of 32 (100000) by mutation

since too many bits have to be changed. Alander and Lampinen [1] compared a 'real-coded' GA with a 'binary-coded' one on an optimization of an internal combustion engine valve cam. They conclude that while the two algorithms showed non difference in terms of function evaluations required for convergence, the time needed to encode and decode the binary representation, doubled the elapsed time for an optimization, from three to six hours. Furthermore, it has been observed that the closer the genotype representation matches the respective phenotype, the more effective the EA is at traversing the decision variable space. The algorithm developed by Leyland [46], and implemented in this work, uses real-value representation of all decision variables.

The next step in the implementation of an EA is to define a "fitness" function for the optimization problem. The fitness function will be evaluated for each individual to determine how well it performs in the population. Fitness may be a direct mapping to the individual's objective function value, or it may be modified in some way so that selection not only prefers individuals with low objective function values, but also preserves other characteristics of the population such as diversity. Since EAs do not have an explicit method for handling optimization constraints, the fitness function may be assigned this task by penalizing individuals that violate constraints, thus making them less desirable.

Once the fitness function has been defined, the algorithm is ready to start optimizing. An initial population of individuals is created, and their fitness scores are calculated. After all processing (i.e. ranking, sorting) has been performed on the current population, individuals are selected from the population for mating. A number of parent selection techniques are available for this task. In the *roulette wheel* selection method [31], each solution in a population has a slice of a roulette wheel. Better solutions have proportionately larger sections of the wheel and therefore there is a greater chance they will be selected. If one wants to individuals to become parents, the wheel is spun twice [14]. *Restricted tournament selection* picks two or more individuals from a population and uses a tournament strategy to select the strongest amongst them [37]. Other, less stochastic, methods include selection by rank and selection by fitness. In selection by rank, a member of a population with a rank k will have a probability of being selected proportional to its rank,

$$P_k \propto \frac{1}{k} \tag{2.8}$$

In selection by fitness, an individual will have a probability of becoming a parent equal to its fitness

normalized over the sum of the fitness of the entire population,

$$P_k \propto \frac{F_k}{\sum_{j=1}^{n} F_j} \tag{2.9}$$

While non-stochastic methods provide faster convergence to a solution, they have the inherent disadvantage of not preserving diversity and therefore have no guarantee that the final solution is in fact optimum. Finally, *elitism*, although not really a strategy, is where the best individuals in the population become parents of the next generation.

Crossover operators are used on selected parents to create new individuals. They can operate on their real-value or binary-encoded representation. Many operators have been created for EAs, some of which will now be discussed. *Single point crossover* is one of the oldest methods and operates on individuals in their binary representation [31]. It involves uniformly selecting an integer position k along a binary string (the individual) at random between *1* and the string length less one, $[1, l-1]$. Two new strings (or individuals) are created by swapping all the characters between positions $k+1$ and l inclusively. *Path relinking*, a binary operator, creates a sequence of children between a set of parents *P1* and *P2*, where the first child is a neighbour of *P1* and each subsequent child is a neighbour of the previous child, with the last child also being a neighbour of parent *P2*. In *real variable uniform crossover* [2], a "child" may take a real value for a particular parameter from either of its parents. *Blend crossover*, first proposed by Eschelman and Schaffer [25], takes two parents and creates a new individual in a hypercube surrounding the two parents. *Linear crossover* is a real variable operator where the child is placed on a line between two parents. Bäck [2] named this operator 'generalized intermediate combination' and provides a detailed description. *Simulated binary crossover* operates on real variables and gives similar results to those that would be obtained if the parents were binary encoded and binary single crossover was performed. Deb and Agrawal [23] discuss this operator in detail. Once a new individual has been created, it can join the population immediately or it can undergo mutation on its variables prior to joining.

The purpose of the mutation operator is to preserve population diversity. Too little mutation leads to a loss in population diversity and the increased risk of the optimization solution getting trapped in some local optima. While frequent use of the mutation operator results in slower convergence of the population to an optimum solution, since it undermines the selection of fittest individuals. In binary encoded variable representations, mutation involves the flipping of a single bit from 0 to 1

or 1 to 0. *Mutation rate*, P_m, is used to express the probability that an individual will undergo mutation. A number of traditional global and local mutation operators exist, and the reader is referred to Goldberg [31] for a detailed discussion of such operators. QMOO, the algorithm implemented in this work, uses three different mutation operators on a variable's real-value representation. The reader is referred to Leyland [46] for a detailed description of each one. Having one (or more) of its decision variables undergo mutation, an individual is then ready to join the main population.

Individuals with some measure of their fitness are added to the main population, while individuals found to be inferior are removed from the population. The process of creating new individuals, evaluating their fitness, mating them, and adding or removing them from the main population is repeated many times until some termination criteria is reached. During this process, the population naturally evolves to a set of optimum solutions to the design problem at hand. Popular criteria used in algorithm termination are: (i) some critical number of objective function evaluations, (ii) mean deviation in performance of individuals in the population falls below some specified threshold (i.e. genetic diversity in population is low), (iii) stagnation – marginal improvements in fitness of individuals from one generation to the next, and (iv) some particular solution in the design space has been reached.

EAs have a number of advantages over other optimization techniques. First, EAs are extremely robust and rely little on the existence of derivatives in the design space. This makes them especially suitable for real-world engineering problems, which have design spaces fraught with discontinuities and multiple nodes. Second, EAs are stochastic in nature and require little information about the problem they are optimizing. This makes EAs suitable for a wide range of optimization problems. Finally, EAs such as QMOO, the algorithm implemented in this thesis, can be designed to handle optimization problems with multiple design objectives. For a complete review of multi-objective evolutionary algorithms, the reader is referred to Coello Coello [16; 17] and Deb [22].

One disadvantage of EAs is that they can be computationally intensive, especially when used on multi-objective optimization problems with a large population size.

Introduction of the fundamentals of EAs is now complete. In literature, EAs are further subdivided into three families: (i) Evolution Strategies, (ii) Evolution Programming, and (iii) Genetic

Algorithms. Evolution Strategies (ES) were developed by Schwefel [62] in 1964-1965. Evolution Programming (EP) was developed by Fogel [29] in the early 1960's as a method of designing state machines for predicting sequences of symbols. Genetic Algorithms (GA) are the last, and probably best known, of the three families of EAs. The modern concept of the GA is based on work done by Holland [36] in the early 1960's, and later by Goldberg [31]. For a comprehensive review of each family, the reader is referred to the respective authors.

2.5 Multidisciplinary Optimization

Large-scale product development projects present a set of complex, often conflicting, design requirements, which require the use of formal and structured approaches to design analysis and optimization [43]. Simulation based design involving integration tools, and multidisciplinary optimization (MDO) procedures in conjunction with well established CAD and CAE tools provide for such structured approached to product design. System integration tools were discussed in §2.3, while the present discussion will focus on the topic of MDO.

Sobieszczanski-Sobieski [65] provides a formal definition of MDO:

> Multidisciplinary Design Optimization (MDO) embodies a set of methodologies which provide a means of coordinating efforts and possibly conflicting recommendations of various disciplinary design teams with well-established analytical tools and expertise. MDO involves multiple disciplines, engineering, business and program management, often with multiple, competing objectives. These disciplines may just be analysis codes, which contain a body of physical principles, or, in addition, they may posses some intelligent decision-making capabilities.

This section will present a number of formal MDO methods. Also, existing commercial MDO packages will be reviewed, which provide services similar to the optimization tool developed in this work.

2.5.1 Formal MDO Methods

Formal MDO methods are intended for synthesis of generic, multidisciplinary engineering systems, such as an aircraft or an automotive vehicle, whose design is governed by multiple disciplines

[67, 3]. The key concept behind these MDO methods is a decomposition of the design task into subtasks performed independently in each of the modules, and a system-level or coordination task giving rise to a two-level optimization [42]. Several MDO methods will now be discussed.

All-in-One Method

The all-in-one method, also referred to as Multidisciplinary Feasibility (MDF) in Cramer et al. [19], is the most common approach to MDO problems. In this method, a vector of design variables $\boldsymbol{x}$ is provided to the coupled system of disciplines and a complete multidisciplinary analysis is performed with the value of $\boldsymbol{x}$ to obtain the system output variable $U(\boldsymbol{x})$, which is then used in evaluating the objective function $F(\boldsymbol{x}, U(\boldsymbol{x}))$ and the constraints $g(\boldsymbol{x}, U(\boldsymbol{x}))$.

The optimization problem becomes:

$$\text{minimize:} \quad F(\boldsymbol{x}, U(\boldsymbol{x})) \tag{2.10}$$

$$\text{subject to:} \quad g_i(\boldsymbol{x}, U(\boldsymbol{x})) \leq 0, \qquad i = 1, \ldots, l, \quad \boldsymbol{x} \in \mathcal{S} \tag{2.11}$$

$$x_i^l \leq x_i \leq x_i^u, \qquad k = 1, \ldots, n \tag{2.12}$$

Individual Discipline Feasible (IDF) Method

The IDF method allows the optimizer to drive individual discipline models to multidisciplinary optimality by controlling the interdisciplinary coupling variables. Variables that represent communication, or coupling, between different design models are treated as optimization variables and are in fact indistinguishable from design variables from the point of view of an optimizer solving a single design model.

Using the IDF formulation, the optimization problem becomes:

$$\text{minimize:} \quad F(\boldsymbol{X_d}, U(\boldsymbol{X})) \qquad w.r.t. \quad X = (X_d, X_\mu) \tag{2.13}$$

$$\text{subject to:} \quad g_i(\boldsymbol{X_d}, U(\boldsymbol{X})) \leq 0, \qquad i = 1, \ldots, l, \quad \boldsymbol{x} \in \mathcal{S} \tag{2.14}$$

$$C(\boldsymbol{X}) = X_\mu, \qquad where \quad \mu = 0 \tag{2.15}$$

$$X_i^l \leq X_i \leq X_i^u, \qquad k = 1, \ldots, n \tag{2.16}$$

In the above formulation, $\boldsymbol{X_d}$ is a set of design variables and $\boldsymbol{X_\mu}$ is the set of interdisciplinary coupling variables. C is referred to as the interdisciplinary constraint. For a detailed description of this MDO method the reader is referred to Cramer et al. [19].

Collaborative Optimization (CO)

The CO formulation is intended for solving MDO problems with loose couplings between analyses with individually large dimensions. It is a two-level hierarchical scheme for MDO, with the top level being the system optimizer that optimizes on the multidisciplinary variables while minimizing the system design objective. There are also optimizers at the individual discipline level, and they attempt to minimize, in a least square sense, the discrepancy between individual discipline design variables, and their system level values, while minimizing the design function of that subsystem.

For a detailed discussion of this method, the reader is referred to Braun and Kroo [6].

Concurrent Sub Space Optimization (CSSO)

CSSO, introduced by Sobieszcznski-Sobieski [68], is a non-hierarchic system optimization algorithm that optimizes decomposed subspaces concurrently, followed by a coordination procedure for directing system convergence and resolving subspace conflicts. This corresponds to common design practice where individual design teams optimize their local component designs, while interdisciplinary compromises are made at the system integration level.

For a detailed discussion of this method, the reader is referred to the original work.

Bi-Level Integrated System Synthesis (BLISS)

The BLISS method, introduced by Sobieszczanski-Sobieski et al. [66], uses a gradient-guided path to reach the improved system design, alternating between the set of disciplinary problems and the system level design space. The general system optimization problem is decomposed into a set of local optimization dealing with a large number of local design variables and a system level optimization dealing with a relatively small number of global variables. Solution to a system level

problem is obtained using either (i) the optimum sensitivity derivatives of the local variables with respect to the system level design variables and the Lagrange multipliers of the constraints obtained at the solution of the disciplinary optimizations, or (ii) a response surface constructed using either the system analysis solutions or the subsystem optimum solutions.

A detailed description of this method is included in the reference mentioned above.

2.5.2 Commercial Applications Providing MDO Services

All engineered and manufactured systems, such as automotive vehicles or aircrafts, experience various interactions between the different components in the system. Taking advantage of these interactions is the mark of a good design, but understanding them is a difficult task. Advances in high performance computing, have made MDO methods popular engineering tools used for this task. Today, a number of independent software vendors, provide system integration tools with MDO problem solving capabilities. This section will present several, commercially available, MDO packages[9].

iSight™

iSight™, developed by Engineous Software™[10], is a comprehensive design integration tool offering an extensive list of optimization techniques and algorithms. It uses a procedural workflow to integrate design models, to express what solving needs to be performed, and in what order [10]. The problem solving mechanism is a centralized controller, which dispatches work items in some predetermined order. iSight™offers a variety of deterministic and stochastic optimization algorithms. It can also be used as a plugin to other third party applications, such as Excel™or Matlab™.

ModelCenter™

ModelCenter™, developed by Pheonix Integration™[11], is a system integration tool with MDO capabilities. Models from third party applications communicate with each other through soft-

[9]A complete list of existing MDO products can be found at: http://www.sgi.com/industries/manufacturing/mdo/
[10]Company website: http://www.engineous.com/index.htm
[11]Company website: http://www.phoenix-int.com/

ware wrappers, which convert model inputs and outputs into a standard format for the ModelCenter™integration environment. The ModelCenter™"Scheduler" serves as a central controller, which determines the execution process based on some pre-defined workflow [10]. ModelCenter™supports a variety of optimization algorithms.

Adaptive Modeling Language (AML)™

AML is a knowledge-based engineering modeling framework developed by Technosoft™[12]. Key features of AML include dependancy tracking, demand-driven calculations, run-time model modification and collaborative engineering [75]. The demand-driven calculation approach dictates that for a given change in some variable, only calculations dependent on that parameter will be performed, but the entire model will not be re-solved [10]. This is an effective approach when the models are large and take a long time to compute.

AMOpt™is a suite of tools for performing optimization and probabilistic design studies within the AML application. AMOpt™includes a variety of deterministic and stochastic optimization algorithms, including: (i) multi-objective genetic algorithms, (ii) sequential quadratic programming (SQP), and the (iii) Nelder-Mead simplex method. AMOpt™can also perform DOE, Monte Carlo simulations and analyses using the response surface methodology (RSM).

2.6 Summary

Product development is the process of transforming a particular costumer need to a final product or service that addresses that need. It is a multidisciplinary activity requiring coordination from all functions of an organization. The size and scope of most product development projects poses a number of challenges, which make this a difficult task. Two specific challenges have been identified in this work: (i) the ability to create an integrated product model, and (ii) the ability to resolve design tradeoffs during product development.

To address the first challenge, research teams and commercial enterprises often employ commercial software packages that allow for some level of integrated product modeling. If successful,

[12]Company website: http://www.technosoft.com/

such tools offer a streamlined development process, accelerating the time to market and improving product quality. A number of product integration tools have been discussed in this chapter, including tools for data sharing, task management, collaborative engineering and conflict resolution. Although the integration tools employ a wide variety of architectures, data models and communication technologies; they all rely on some form of a consolidated, explicit description of the complete integrated system model. DOME, an integrated modeling environment developed at the MIT CADLab, does not require an explicit system definition of a central workflow model the solving of all relationships in the integrated system model. Distributed models inter-operate at a global level, but maintain control at a local level. This architecture allows the system to emerge and evolve rapidly.

Numerical optimization techniques can be an effective tool in resolving conflicting design objectives during product development. Several optimization techniques were presented in this section, including dynamic programming, linear optimization, calculus-based, enumerative and heuristic. Discussion of heuristic algorithms focused on two types: simulated annealing and evolutionary. The latter was discussed in greater detail, since the algorithm implemented in this work is evolutionary.

MDO frameworks extend numerical optimization theory to provide a structured approach to design analysis and optimization. They empower designers to better understand complex, often conflicting, design requirements so that optimal design tradeoffs can be made. Many approaches to MDO have been developed and some have been discussed in this section. With advances in high performance computing, many independent software vendors are developing MDO packages capable of successfully evaluating complex engineering systems. Several commercial MDO packages have been reviewed in this section.

The generic product development process and various system integration environments have now been introduced. Also, numerical optimization and its implications on multidisciplinary design havef been discussed. The next chapter will present QMOO, a multi-objective evolutionary algorithm, developed by Leyland [46] and employed in the target-driven design tool developed in this work.

Chapter 3

The Queuing Multi-Objective Optimizer

3.1 Introduction

The queuing multi-objective optimizer (QMOO) was developed by Leyland [46], at the Laboratoire dEnergetique Industrielle at the Ecole Polytechnique Federale De Lausanne. Initially developed for optimizing energy system problems, QMOO was employed to develop the target-driven design tool because it best suited the DOME computational environment. This section will describe QMOO in detail. First, the criteria that were considered in selecting an optimization algorithm will be presented. Next, important design features of QMOO will be summarized, which have had a profound influence on the algorithms design. The discussion will then extend to describe the QMOO algorithm and its approach to problem optimization. Finally, the topic of parallelism in QMOO will be introduced.

3.2 Criteria For Algorithm Selection

To effectively optimize a DOME-enabled system model of any product, the optimization algorithm should account for and be able to handle the following scenarios:

- Since designers often use non-trivial CAE tools (ex. FEA) to assess the performance of a product, the optimization algorithm must be able to handle non-linear and discontinuous system models. Discontinuities in the optimal surface present serious problems for conventional optimization techniques, such as gradient-based searches. As a result, a heuristic optimization algorithm was preferred.

- Given that a designer will most likely measure the performance of a product design based on more than one design objective, the optimization algorithm must be able to handle multi-objective problems.

- Engineers often make use of complicated non-linear models that take minutes, sometimes hours of computation to solve. Given that the optimization algorithm will encounter objective functions that will take a long time to evaluate, the algorithm should be designed with an architecture that lends itself to parallelism. Parallelism will be discussed in later chapters.

The QMOO algorithm was selected for the target-driven design tool because it satisfied all of the criteria mentioned above. Furthermore, the algorithm was tested for convergence to the Pareto-Optimal Front (POF), and it was found to perform better than existing algorithms [46]. QMOOs rapid and robust convergence can be attributed to its strong elitism and a number of innovative features, such as tail preservation and the evolutionary operator choice mechanism. These and other features of the QMOO algorithm will be described in the following section. For a detailed discussion of QMOOs performance, the test approach and test problems, the reader is referred to Chapter 5 of Leyland [46].

3.3 Design Features of the Queuing Multi-Objective Optimizer

This section will discuss a number of important design features that have had large influence on the QMOO algorithm described in the next section. Implementation of these design features was driven by the goal of creating an optimization algorithm that can provide rapid and robust convergence to as many local optima as possible [46].

Important design features of QMOO are:

- *Steady state.* QMOO is a steady-state evolutionary algorithm. This means that the population created by this algorithm has no generational structure. Individuals are added to the population in a process that is completely separate from their removal and there is no formal control of population size.

- *Elitism.* QMOO is an extremely elitist algorithm. This means that it has a single population, which contains only the best individuals found so far. The advantage of this approach is that

the algorithm converges rapidly, however the disadvantage is that it poses problems for the preservation of diversity and could limit the algorithms ability to explore the design space by converging on some local optima. This diversity preservation problem lead to some interesting developments to ensure that the algorithm remained elitist. The most significant of these efforts is the dividing of the population into groups.

- *Grouping.* QMOO preserves diversity by dividing the population into groups in the parameter design space, and then letting each group evolve independently. Groups do not compete, but occasionally individuals can be created by breeding between groups. Grouping of the population from time to time can be a time consuming task and this must be allowed for in the QMOO algorithm. The author argues that QMOO was designed for optimization of problems, where the objective function can take seconds, sometimes minutes, to solve. Therefore, any complex analysis of the population that can reduce the number of objective function evaluations needed for convergence is probably worthwhile.

- *Multi-Objective Optimization.* QMOO implements Pareto-based multi-objective optimization. This requires the use of computationally expensive population ranking algorithms to determine which individuals are non-dominated, and therefore Pareto-optimal. QMOO uses non-dominated sorting.

- *Asynchronous architecture.* QMOO is implements a parallel architecture that is asynchronous. Leyland [46] describes two problems with synchronous parallel algorithms. First, all the computers in a network stop at the end of each generation and wait for the master computer to create the next generation. In a multi-objective problem, this implies that a considerable amount of time will be spent ranking the population. Second, if the objective function takes a variable amount of time to evaluator or all the computers do no run at the same speed, much time can be wasted at the end of each generation by waiting. The asynchronous approach eliminates both of these problems.

All of these design features lead to the development of a unique optimization algorithm, suitable for the DOME target-driven design tool. The following section will describe the QMOO algorithm in more detail. This discussion is a summary of a detailed description presented in Leyland [46].

3.4 The Queuing Multi-Objective Optimizer

QMOO is an evolutionary algorithm with a queue-based architecture. This means that the algorithm treats members of a population as independent entities that undergo a number of processes before they can become a solution. The queue-based architecture implies that the algorithm does not operate on the entire population in generations, but stores individuals in queues for each process. Individuals are taken off the queue and processed as needed. The processes that a member of a population must go through, or the life of an individual, are broadly illustrated in Figure 3-1. In the original work, the author discusses many variants of QMOO; the variant discussed here includes complex population analyses techniques such as grouping and ranking. This variant is intended for optimization problems where the objective function takes seconds, sometimes minutes, to solve thus making these complex population analysis techniques justifiable.

Each process in Figure 3-1 will now be discussed.

3.4.1 Assignment of Parameter Values

The main loop in QMOO creates an individual and stores it in a queue where it waits to be assigned. When the assignment process is ready, it takes the individual off the queue and assigns it parameter values. If the algorithm is in its initialization phase, parameter values are chosen at random within the allowed ranges. If initialization of the algorithm is complete, parameter values are assigned by crossover and mutation operators.

The assignment algorithm is detailed in Leyland [46], and will not be described here. However, two features of the assignment algorithm are worth mentioning here. Firstly, individuals are chosen for assignment by selecting the group first, and then the individual inside it. This ensures that all groups converge at the same rate, independent of the population size inside any given group. Secondly, after an individual has been assigned its parameter values, a check is made whether it has the same parameter values as any of its parents. Implementing a duplicate detection mechanism is worthwhile for two reasons: (i) in the case where evaluation of an objective function takes a long time, it saves valuable processor time and (ii) it prevents the loss of diversity by not allowing two (or more) individuals to occupy the same point in the decision variable space. In QMOO, two identical points will not exist in the main population at the same time.

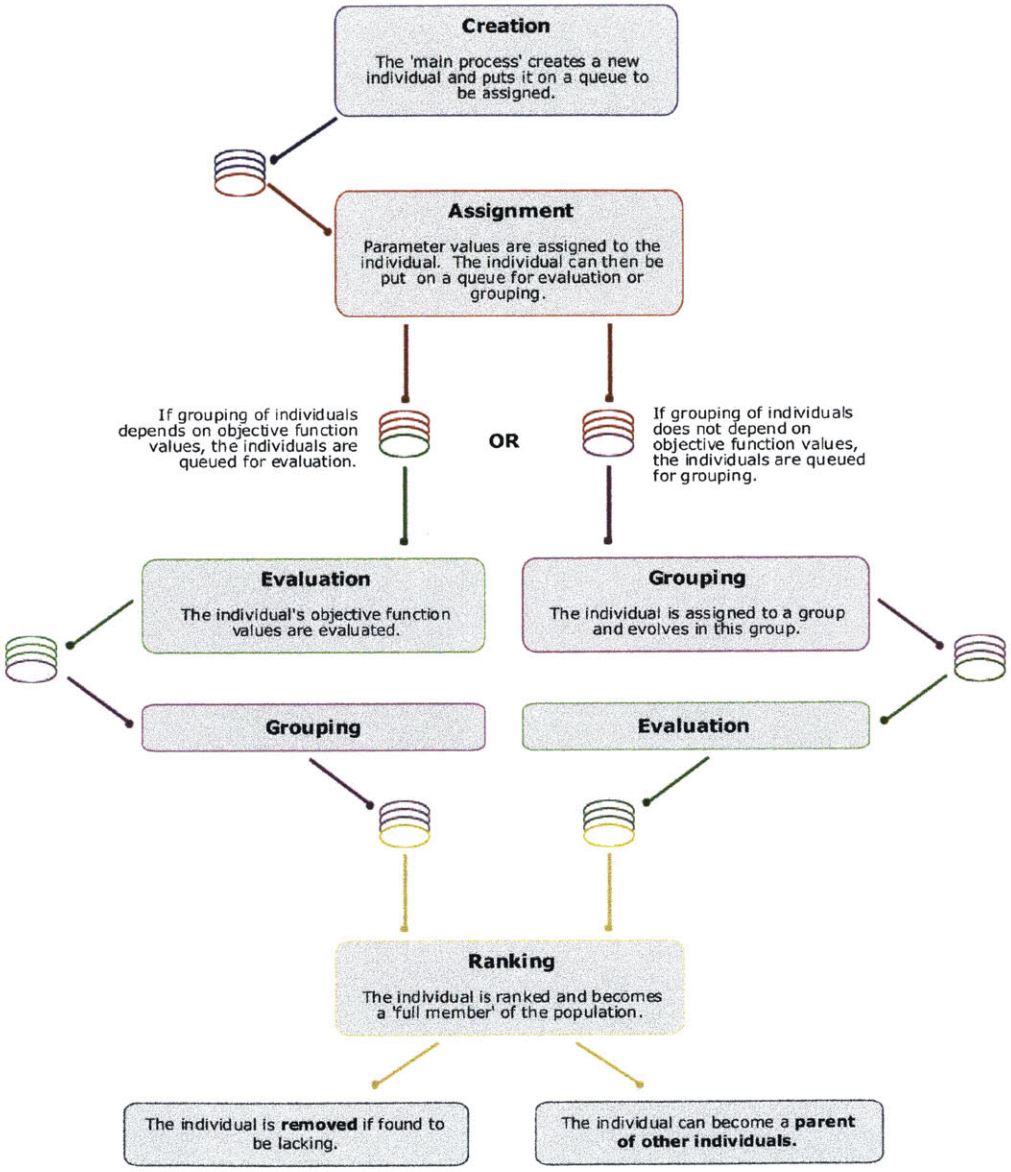


Figure 3-1: The different processes that an individual must go through before it can join the "main population".

QMOO implements a unique evolutionary process to aid in the selection of operators used in assigning parameter values to newly born individuals. This process, named evolutionary operator choice (EOC), allows the choice of combination or mutation operator to evolve with the population. The premise of EOC is that if individuals are generating successful children using a particular operator, those children should continue to use that operator. The success of a child is measured by how long that child remains in the population, thus operators producing non-dominated children will be preferred by the EOC. One has to be cautious when using this approach. Using an operator that causes a population to converge rapidly can lead to diversity problems and limited exploration of the POF. Leyland [46] describes EOC in detail.

Effective mutation operators are of utmost importance in maintaining population diversity. The mutation operators in QMOO were designed to work with EOC. There are three operators in total: (i) normal mutation, (ii) uniform mutation and (iii) global mutation. The operators are described in detail in Leyland [46].

3.4.2 Evaluation of Objective Functions

The QMOO algorithm requires little information about the objective function it is evaluating. As far as QMOO is concerned, the objective function is a complete black box. Decision variable values are passed into it, and objective function values return back to QMOO. Evaluation of the objective function values is trivial. Furthermore, since the individuals needing objective function evaluation are stored in a queue, the algorithm can easily be configured to run optimization function evaluations in parallel.

3.4.3 Grouping

The purpose of using grouping in QMOO is two-fold: (i) to find multiple local optima in the model design space, and (ii) to effectively preserve population diversity. A designer would like to find as many local optima as possible because they represent the most interesting regions of the model. Preservation of diversity is critical in an elitist algorithm, such as QMOO, because it may lead to a more robust convergence resulting in a better approximation of the POF.

In QMOO, the population is grouped in decision variable or design objective space. To preserve diversity, the groups evolve as (almost) entirely separate populations, without competing with each other. However, groups can interbreed and the population is re-analyzed frequently to re-assign individuals to different groups, merge groups, or create new ones.

The grouping algorithm gives groups, not individuals, equal opportunity to breed. This is to ensure that all of them can converge at a similar rate. If individuals were to be given equals chances to breed, it is likely that at some point one group would become larger than the others, evolving quicker and, due to its larger population, breeding more often. Once the non-dominated set of individuals in this large group reaches the POF, fewer individuals in this group will be added in each generation only because they manage to squeeze between two other, top-ranked points in the group. This process will add computationally expensive resolution to the non-dominated set where it is not needed. Concurrently, smaller groups will receive less computational resources to converge towards some other optima, which would give the designer much more valuable information about the model. For this reason, the grouping algorithm in QMOO gives the same number of breeding opportunities to each group.

Individuals are grouped into clusters using fuzzy c-means clustering (FCM). This technique attempts to minimize the distance from each individual to the center of its cluster. It does this in two ways: (i) by moving the centers of the clusters and (ii) by moving points from cluster to cluster. The clustering is fuzzy because points are not assigned exclusively to a specific cluster, but instead have a percentage of membership in each cluster. For example, a point lying in the middle of some cluster may be 95% member of that cluster, and a few percent member of every other cluster. The author chose to use FCM in the QMOO algorithm because it is quick and performs well. For a more detailed description of FCM the reader is referred to Leyland [46].

Grouping of a population leads to the preservation of its diversity, or the preservation of individuals that are not themselves optimal, but contain valuable information about the design objective space. This leads to a slower solution, but improves the likelihood that the population will discover more regions of the model that would be of interest to the designer. Grouping should only be used in situations where it improves the robustness of a solution, i.e. leads to the discovery of as many interesting regions of the model design space as possible. If this is not the case, grouping will only

result in a slow solution to the optimization problem and should not be used.

3.4.4 Ranking

Ranking is used to determine which individuals in a given population are the fittest. In QMOO, the ranking process is responsible for ranking members of a single group and requires only a list of objective function values for each individual in that group. The ranking algorithm, and the particular dominance matrix used to sort individuals, is described in detail by Leyland [46]. This section will focus on one particular process managed by the ranking algorithm, namely, tail preservation.

Tail preservation is a technique used to explore single POFs. When the population is entirely non-dominated (i.e. has a high number of first-rank individuals), it is difficult for inferior (i.e. second, third, -ranked) individuals to enter the final population. Given that the non-dominated set will likely converge to a section of the POF, as in Figure 3-2, some second-ranked individuals will fall in the tail region of the non-dominated set. These points represent valuable information about how to find the remaining parts of the POF. Tail preservation identifies these important individuals and keeps them in the population to explore the full span of the POF.

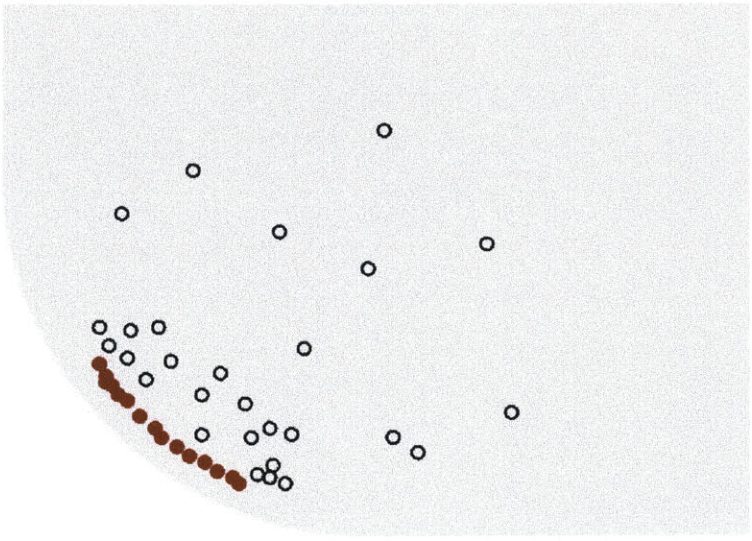


Figure 3-2: Dominated individuals (un-filled points) contain valuable information about the full range of the POF.

Tail boxes are defined at the ends of the NDS (Figure 3-3) and have 5% of its linear dimensions.

Individuals that are not in the first rank are examined to see if they fall into the tail regions. Typically, the tail box region contains one or two individuals, which is not so large that time is wasted on sub-optimal points. The tail regions are ranked in exactly the same way as the non-dominated set, except that for each region, one of the objective directions has been reversed. For example, for the box on the lower right, the fittest individuals would have a minimum value on the vertical axis and a maximum value on the horizontal axis. These nested optimization problems results in a robust population convergence that covers more of the POF.

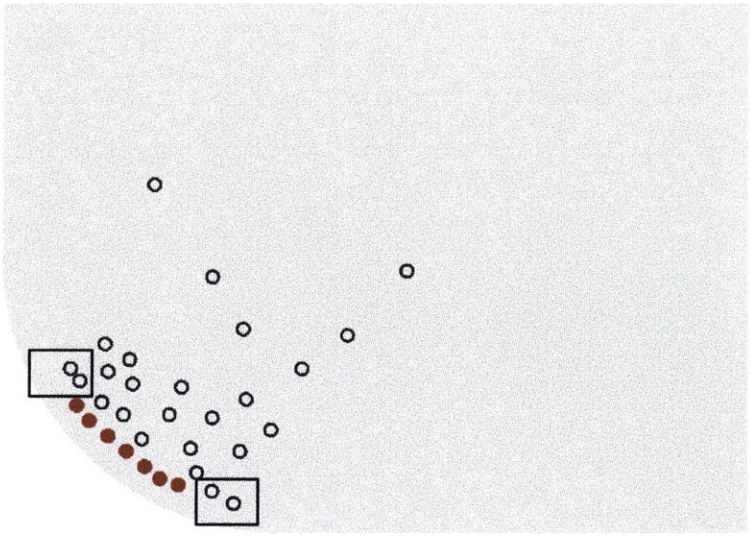


Figure 3-3: Preservation of "tail regions". Individuals in the boxes at the ends of the NDSS are preserved, and ranked preferring "left and up" (left box) and "right and down" (right box) [46].

QMOO currently supports tail preservation for two objectives only. In two-objective space, the tail regions can be easily defined with boxes. Extending this to optimization problems with more than two objectives presents some complications as tail regions become awkward to define.

3.4.5 Parent Selection

QMOO implements a parent selection method that is based on the rank of an individual in the current group. Higher ranked parents are preferred, but this can be turned off completely using a tuning parameter. According to the author, this method was chosen for its versatility, rather than any theoretical reason, and will be discussed below.

When the population contains several ranks of individuals, the proportion of each rank in the population and the cumulative probability of an individual being in rank n or better are calculated. If no preference to rank is desired, a random number is chosen from a uniform distribution between 0 and 1, the corresponding rank is chosen and a random individual is chosen from that rank. If the random number is raised to a power greater than 1 before choosing the rank, parent preference can be controlled by the exponent.

3.4.6 Thinning

QMOO is an elitist algorithm and will therefore spend much of its time with an entirely non-dominated population. When the algorithm reaches this phase, ranking of individuals can no longer be used to provide selection pressure and the size of the group (and population) can grow unchecked. To ensure that the algorithm can still perform well, some of the non-dominated individuals from the group must be removed (i.e., the group must be thinned out). Individuals best suited for thinning, are those that offer little information about the form of the POF (i.e., individuals in crowded regions of the NDS) or individuals that are far from the POF.

Two thinning techniques were developed in QMOO and both maintain selection pressure in the population, and provide even coverage of the POF. The first method, dominated volume thinning, removes individuals that add the least to the volume dominated by the NDS. The concept of non-dominated volume was introduced by Zitzler [80] and is a measure of the performance of an algorithm the larger the volume dominated by the NDS generated by the algorithm, the better the algorithm. The second method, quadratic thinning, chooses a subset of the NDS and fits a least-squares quadratic through this subset. Individuals dominated by the quadratic become candidates for removal. The choice of the subset over which to interpolate and of which individuals dominated by the quadratic to remove are made such that the group, or population provides an even coverage of the POF. Dominated volume thinning and quadratic thinning are discussed in detail in Leyland [46].

3.5 Parallelism in the Queuing Multi-Objective Optimizer

The queue-based algorithmic structure of QMOO makes its parallelism quite flexible. Instead of an individual having its objective function evaluated immediately after it has been assigned its decision

variables, the individual is put on a queue of individuals waiting to be evaluated. Any number of available slave computers can pick an individual off the end of a queue, evaluate it, and return the individual, which now has an objective function value, back to the master computer. This is the flexible queue-based approach to parallelization in QMOO.

Running an optimization in parallel fashion over a network of computers requires that the algorithm be able to handle a number of possible complications, such as network latency, computers of different speeds or crashing of simulations. The parallel architecture in QMOO attempts to alleviate these potential problems with the following approach:

- *Network latency.* Each slave processor keeps its own queue of individuals to be evaluated. This is to ensure that while an individual is being transferred, the processor can being the next individual in its queue.

- *Shutting down slave computers.* The master computer checks to that the slave computers are returning individuals passed to them. If the slave computer is not responding, it assumes that the slave computer has been shut off and the individuals waiting to be evaluated are passed to other processors.

- *Computer crashes.* If the individual is passed to a slave computer more than three times and never returns with the objective function evaluated, the master assumes that this individual is causing the slave to crash and the individual is marked as infeasible.

Experiments with parallelism in QMOO, performed by Leyland [46], showed that solving large problems using this approach reduced the elapsed time by a factor of about four on a network of six computers. The author also experimented with parallelizing ranking and grouping, however it was noted that due to high network latency this had very little effect on the performance of the algorithm. The author proposes that dedicating a computer to perform each specific task could remedy this problem. However, this was not experimented with.

3.6 Summary

QMOO is a multi-objective optimizer and is the optimization engine of the DOME-enabled target-driven design tool. Its efficient, queue-based design and suitability for design objectives that take a

long time to evaluate, fits well with the DOME computational environment.

Early in the development of the target-driven design tool, a number of optimization algorithm requirements were identified. The algorithm must be able to optimize non-linear, even discontinuous, system models. The algorithm must be a multi-objective optimizer empowering designers to elucidate and evaluate design tradeoffs. Finally, given that in large-scale system models design objectives can take a long time to evaluate, the optimization algorithm should support parallelization. QMOO, the algorithm selected for this work, satisfies all of the above requirements.

QMOO is a steady state, multi-objective, evolutionary optimizer developed at LENI for optimizing energy problems. QMOO is also an extremely elitist algorithm, which means that it has a single, non-generational, population that contains only the best individuals found so far. The advantage of this approach is rapid convergence. However, having only the best individuals in the population leads to problems with diversity preservation. Two techniques have been developed to overcome this obstacle. The first technique, clustering of the population into groups, is intended to provide robust convergence to multiple local optima. The second mechanism, tail preservation, is meant to improve the exploration of single Pareto-optimal frontiers. Finally, QMOO uses a unique operator, termed the Evolutionary Operator Choice (EOC), for creating new individuals in the problem. EOC means that the best operator for a given problem will be used without any input or tuning from the user.

The queue-based architecture of QMOO makes it quite flexible towards parallelism. QMOOs asynchronous parallel architecture results in an efficient use of computational resources, when delegating objective function evaluations to slave computers across a network. Finally, the algorithm design accounts for a number of potentially problematic scenarios: (i) network latency, (ii) shutting down of slave computers, and (iii) computer crashes.

The introduction of the QMOO algorithm thus complete, the discussion will now proceed to introduce the DOME-enabled optimization tool that makes use of it. The following chapter will also discuss how this design tool empowers decision makers to practice a target-driven design approach.

Chapter 4

Target-Driven Design

4.1 Introduction

This chapter will describe the DOME-enabled target-driven design tool, developed in this work, which empowers designers to practice an efficient, target-driven engineering design approach. The target-driven design tool is essentially an optimization tool and combines the seamless integration capabilities of DOME with evolutionary optimization techniques, discussed in Chapter 3, to rapidly explore the design space of a product system model and to present the decision maker with an initial set of design candidates, or points on the Pareto-optimal frontier, that best satisfy the specified design objectives. The decision maker can then pick a point on the Pareto-optimal frontier and set the state of the entire integrated product model to reflect the optimal target configuration of the design objectives under examination. This process is illustrated in Figure 4-11.

The discussion in this chapter will begin by introducing a trivial design example, the tube-bundle design scenario. Next, this design scenario will be used to demonstrate the use of the optimization tool. Subsequently, the inner workings of the optimization tool will be explored focusing on the communication mechanism between DOME and the optimization engine discussed in Chapter 3. Finally, the implications of the optimization tool on the generic engineering design process will be discussed.

4.2 The Tube-Bundle Design Scenario

The design scenario is an in-house heat-transfer engineer collaborating with an external parts supplier to determine the optimal tube geometry for a tube-bundle heat exchanger. The engineer is considering two competing design objectives: (i) cooling performance and (ii) cost of fabricating the tube. Since the two objectives are competing, it is impossible to improve the cooling performance of the tube without making it more expensive to fabricate. Scenario participants and their service models are summarized in Table 4.1.

Design Expert	**Service Model**	**Modeling Software**
Heat-Transfer Engineer	•Modifies geometric properties •Selects tubing material •Sets temperature conditions •Calculates cooling performance	DOME-native model
External Part Supplier	•Modifies geometric properties •Selects tubing material •Calculates cost of tube fabrication	EXCEL™spreadsheet

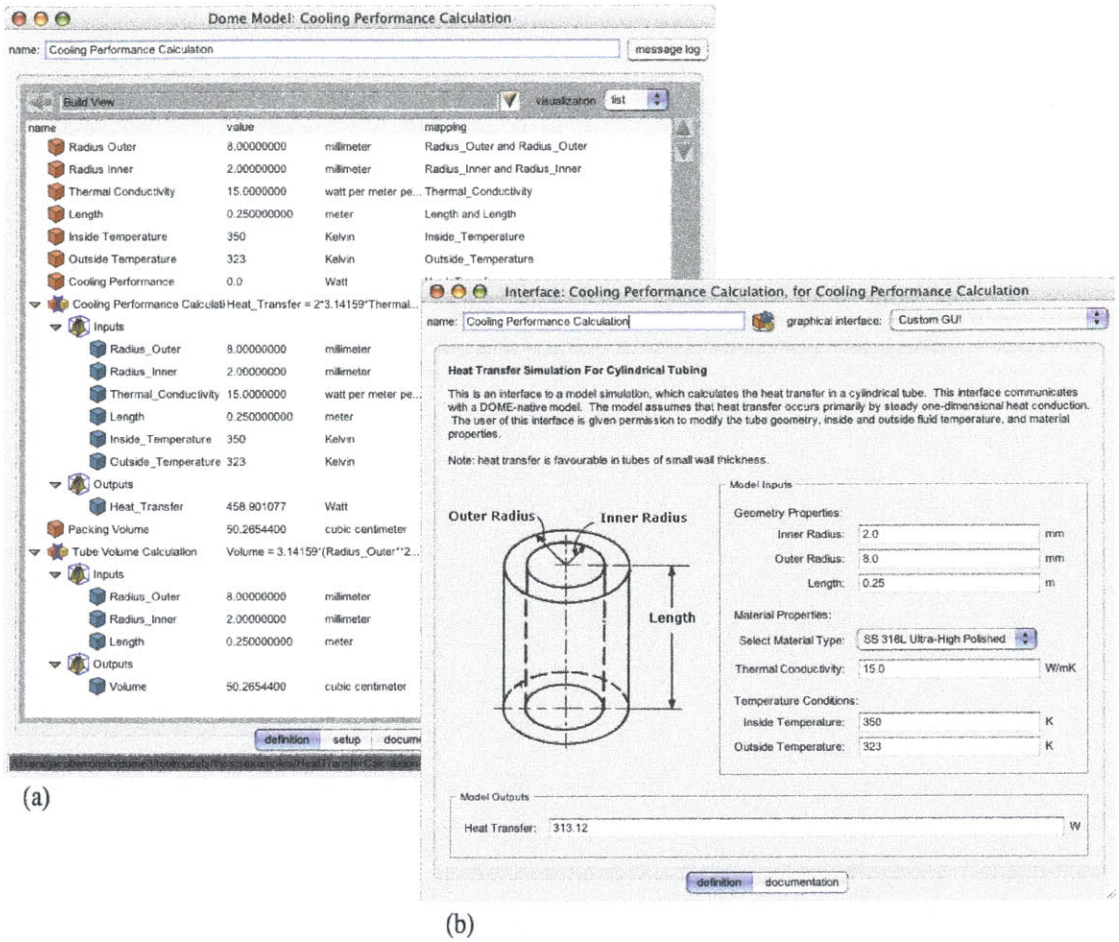
Table 4.1: Description of models and expertise in the tube-bundle design scenario.

4.2.1 The "Heat Transfer Engineer" Expert

The heat-transfer engineer is an expert at modeling cooling performance of heat exchanger components. For this scenario, heat transfer in the tube is modeled using *Fouriers Law* assuming steady, one-dimensional conduction [48]. Figure 4-1 shows a screen image of the DOME-native model created by the engineer for this purpose. The owner of this model allows individuals, subscribing to this heat transfer analysis through the model interface shown in Figure 4-1, to vary the geometric dimensions and material properties of the tube, and the operating temperature conditions. The output of the model to the interface is the amount of heat conducted through the tube.

4.2.2 The "External Part Supplier" Expert

The external part supplier provides material and fabrication cost services for tubing components. Figure 4-2 shows a screen image of an EXCEL™spreadsheet model created for this purpose. The



(a)

(b)

Figure 4-1: (a) DOME-native model for cooling performance calculations in cylindrical tubes, and (b) a DOME custom graphical user interface used to interact with the model.

model requires geometry inputs and material specifications to calculate the total tube fabrication cost. The back-end cost equation is formulated such that, for the interval considered, the cost of fabricating the tube is inversely proportional to its wall thickness. This does not have any theoretical or practical basis, it only satisfies the condition that the two design objectives, cooling performance and fabrication cost, are competing against each other.

4.2.3 The Design Scenario

The heat transfer engineer must collaborate with the external part supplier to optimize the overall tube design, balancing cooling performance and cost of fabrication. This requires some form of

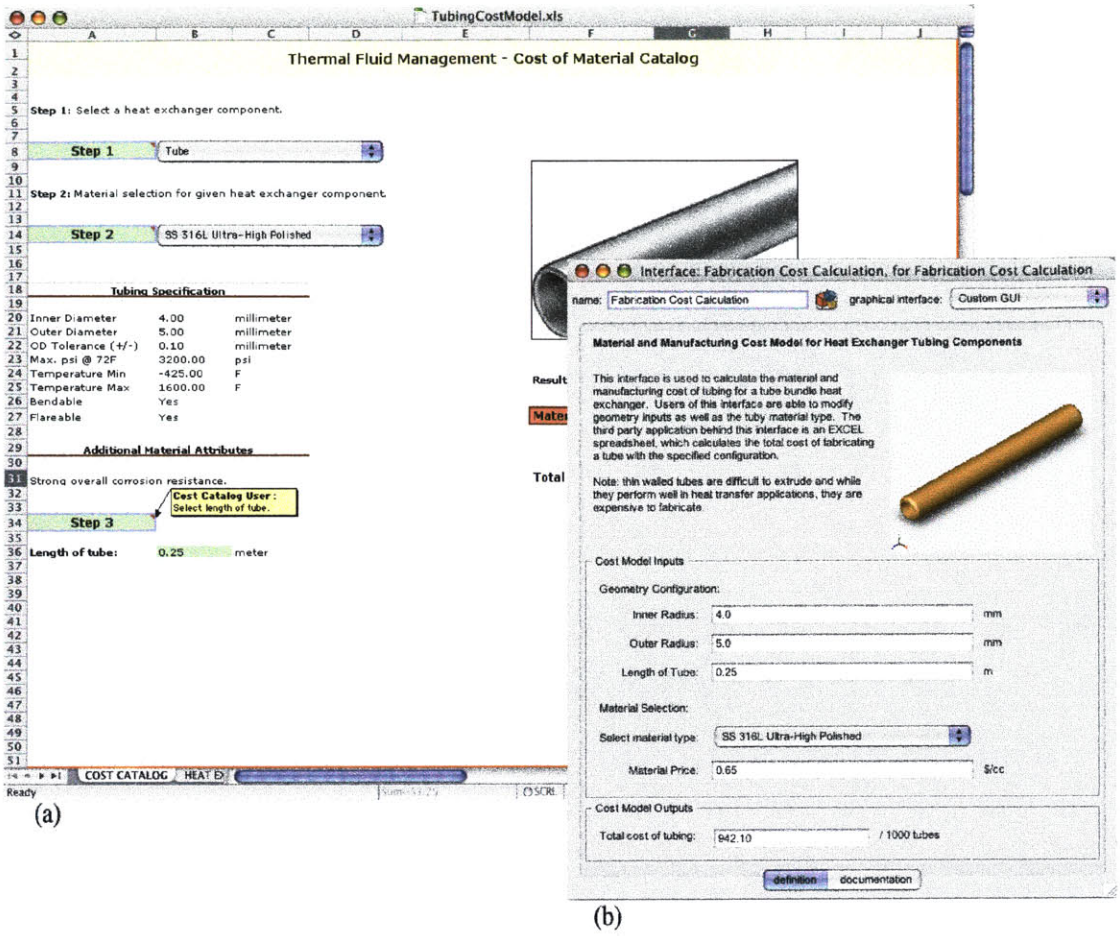


Figure 4-2: (a) EXCEL™-native model for calculating the tube fabrication cost (per 1000 pieces), and (b) a DOME custom graphical user interface used to interact with the model.

model integration between the two experts, so that the engineer is able to identify design alternatives that perform well against both, local design objectives and those defined by the part supplier. Senin et al. [63] state that in a traditional design environment, obtaining such an integrated view, for just one design alternative, is at best very time consuming to coordinate and resolve. At worst, obtaining an integrated system view for making global tradeoffs is deemed intractable and decisions are left to intuition. Clearly such an unintegrated product model will not lend itself well to the exploration of many design alternatives.

4.3 Using the DOME-enabled Optimization Tool

This section will describe the optimization tool developed to facilitate the rapid elucidation and evaluation of design tradeoffs, which are key elements of the target-driven design approach. The discussion will focus on four processes that constitute the target-driven design approach: (i) building an optimization model, (ii) publishing that model on a DOME server, (iii) running a published model to obtain a Pareto-optimal set of solutions, and (iv) evaluating design tradeoffs and picking a target design. To help clarify, the tube-bundle design scenario will be used throughout the discussion.

4.3.1 Building an Optimization Model

The building of an optimization model can be summarized in a four-stage process. First, the model builder creates an integrated product model in the form of a DOME integration project. Next, the model builder defines a set of design variables and objectives, which are parameter objects specific to the optimization tool and correspond to integration project inputs and outputs, respectively. The third stage involves configuring the optimization model and tuning the optimization algorithm. In the final stage, the model builder creates any number of model interface objects. Once these interfaces are published on a DOME server, individual can readily subscribe to them and run the underlying simulations.

Publishing model services on DOME servers will be discussed in §4.3.2.

Creating a DOME Integration Project

The first step in building an optimization model is to create a DOME integration project, which connects all of the individual sub-models through user-defined inter-relationships to form a product system model. Figure 4-3, the heat transfer engineer is adding two resources to the integration project: (i) a heat transfer calculation model published on a local DOME server, and (ii) a remote tube-costing model published, in a different geographical location, at the part suppliers site. Next, the heat transfer engineer must subscribe to the services provided by each model through one of the available model interfaces. The engineer will interact with the models through these subscriptions.

Figure 4-4 shows the heat transfer engineer creating an integration model and subscribing to an

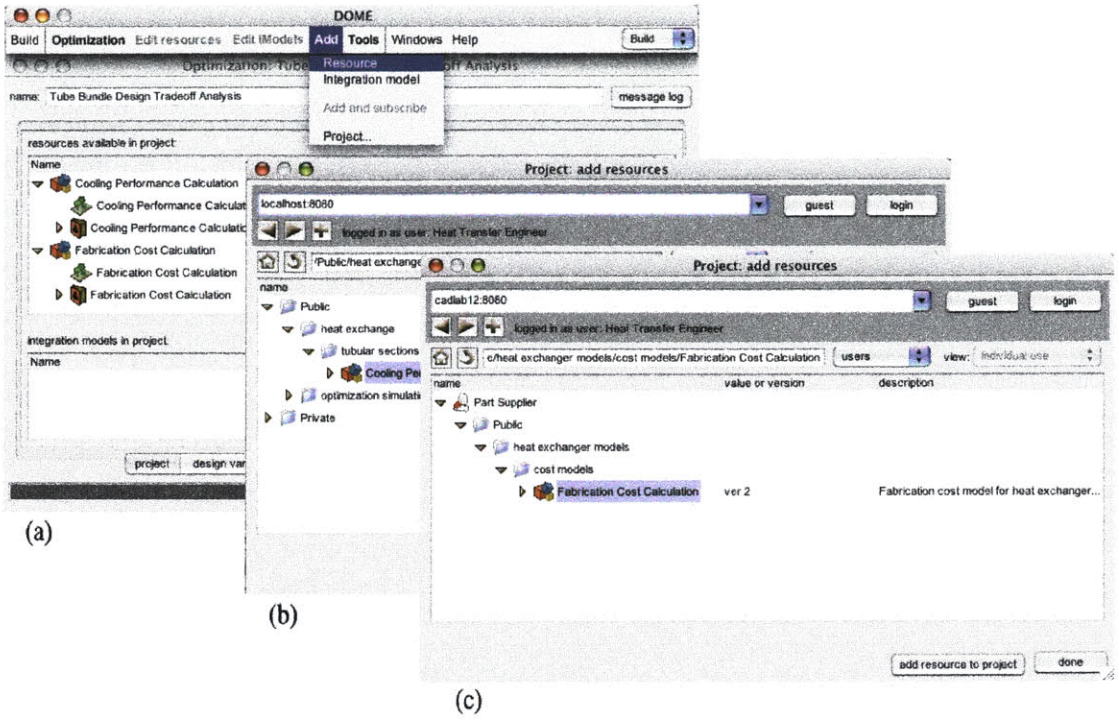


Figure 4-3: (a) DOME project integration panel with two model resources added to it. (b) Adding a local 'Cooling Performance Calculation' resource located on the heat transfer engineer's DOME model server. (c) Adding a remote 'Fabrication Cost Model' resource located on the part supplier's DOME model server.

interface object in each of the added resource models. Once all of the relevant sub-models have been obtained (resource models) or created (integration models), the model builder is faced with the seamless task of inter-relating the individual services to create an integrated product model.

In the tube-bundle design scenario, the heat transfer engineer would like to determine the optimal tube geometry that will maximize cooling performance and minimize fabrication cost. Upon inspection of the two model interface services, it is evident that both of the model calculations are driven by the same three geometric inputs: inner radius, outer radius, and length of tube. The engineer decides to integrate the two models by mapping both sets of model inputs to a newly created set of real number objects. This new set of real number objects is now driving both models, and therefore becomes the set of inputs for the integrated system.

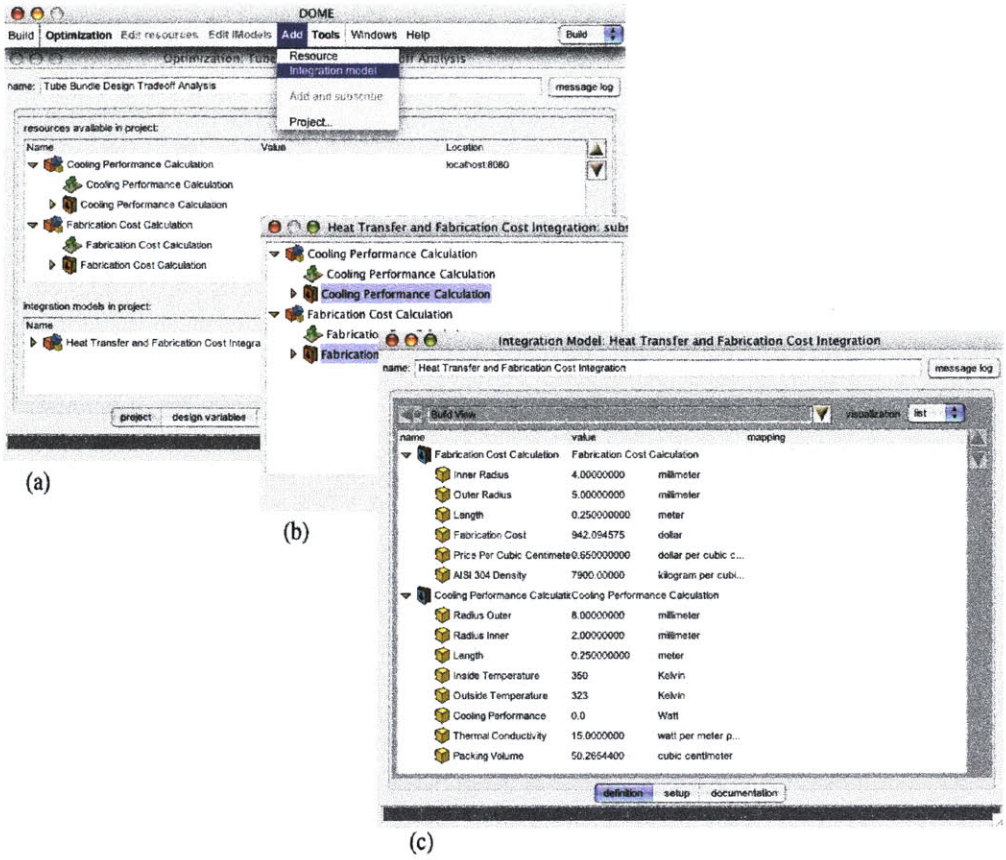


Figure 4-4: (a) The heat transfer engineer is creating an integration model 'Cooling Performance and Fabrication Cost Integration', which will contain the integrated product system model. (b) The engineer is subscribing to each model service through its available interface (high lighted in blue). (c) Expanded view of the integration model, showing the subscribed interface objects and the real number services exposed within them.

Figure 4-5 illustrates this integration with a block diagram where inputs to the models represent decision variables and outputs represent design objectives.

Defining Design Variables and Objectives

The second step in building an optimization model is to define design variables and objectives that the optimization algorithm will use to interact with the model (this interaction will be discussed in detail later in this chapter). Design variables are selected by the user, and they can be any inputs to the DOME integration project. During a solving iteration, the optimization engine will assign a real

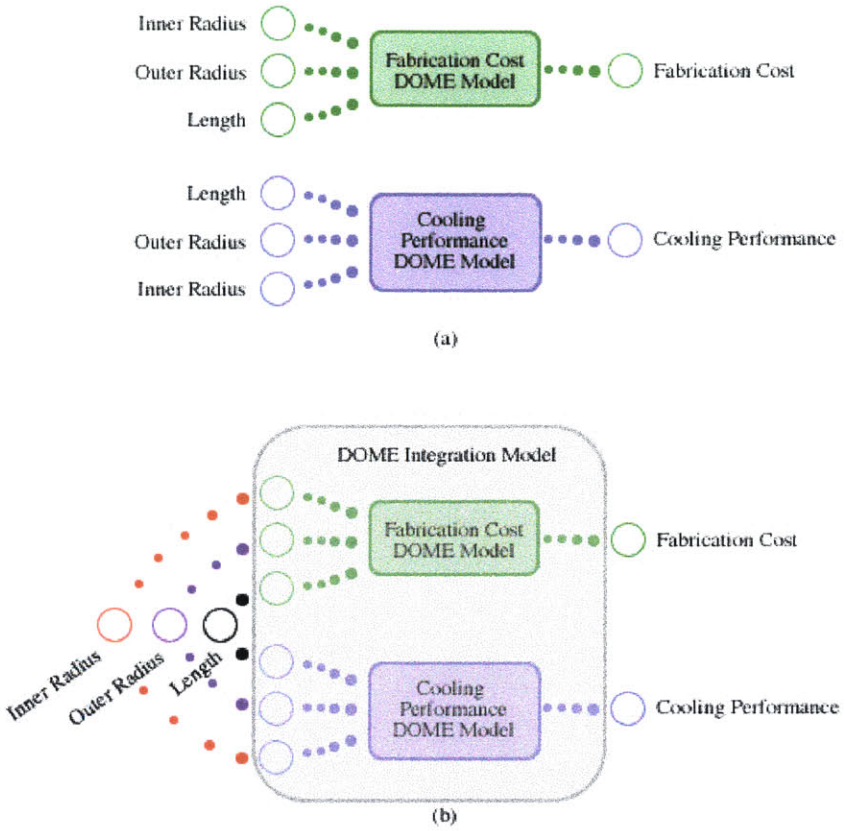


Figure 4-5: (a) Block diagram of the individual DOME service objects in the tube-bundle design scenario, showing model inputs and outputs. Note that both models have the same inputs driving the outputs (b) Inputs of each model have been mapped to a third set of parameter objects. The individual models are now integrated.

value to each input parameter object so that the integration project can be solved and new output parameter objects can be obtained. Variable objects have upper and lower limits, and the optimization engine cannot assign values outside of those limits. Design objectives, also selected by the user, are outputs of the integrated product model. The optimization engine will attempt to minimize or maximize the real values of these objects. In its current form, the optimization tool is only capable of supporting real-value parameter objects. This restriction is imposed by the optimization algorithm, discussed in §3.4, and its future versions should support discrete and discontinuous variables.

Returning to the discussion of the tube-bundle design scenario, the heat transfer engineer may be

interested in studying how different geometry configurations perform with respect to fabrication cost and cooling performance. To proceed with this analysis, the engineer must create the necessary decision variable objects and map them to the relevant dimension inputs in the integrated model, in this case outer radius and length. Furthermore, the engineer must also create two design objective parameter objects and map them to the two outputs, fabrication cost and cooling performance. This process is illustrated in Figure 4-6.

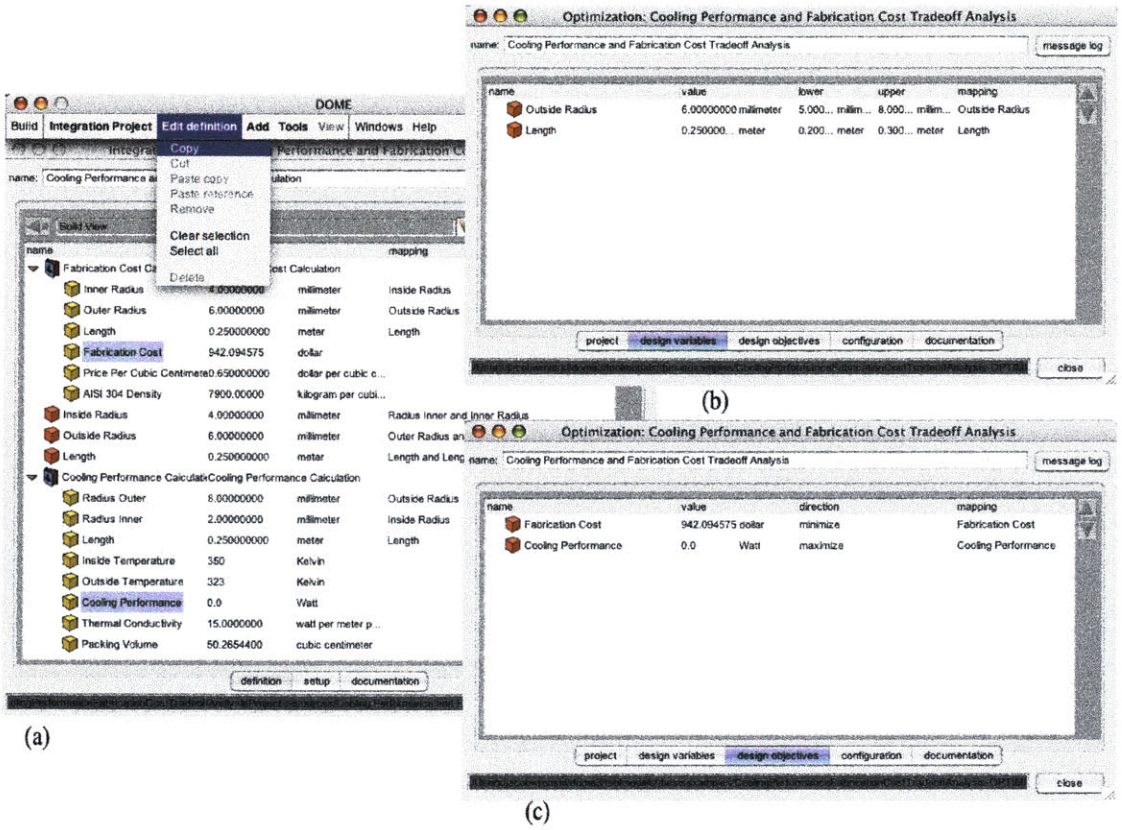


Figure 4-6: (a) Integration model showing two subscription objects and three parameter objects (red icons), the heat transfer engineer has highlighted two parameter objects and is about to map them to the design objectives. (b) 'Outer Radius' and 'Length' have been defined as design variables. (c) 'Cooling Performance' and 'Fabrication Cost' have been added and mapped to the design objectives panel.

Configuring The Optimization Algorithm

The configuration panel of the optimization tool is shown in Figure 4-7 and each of the tuning

parameters is discussed below:

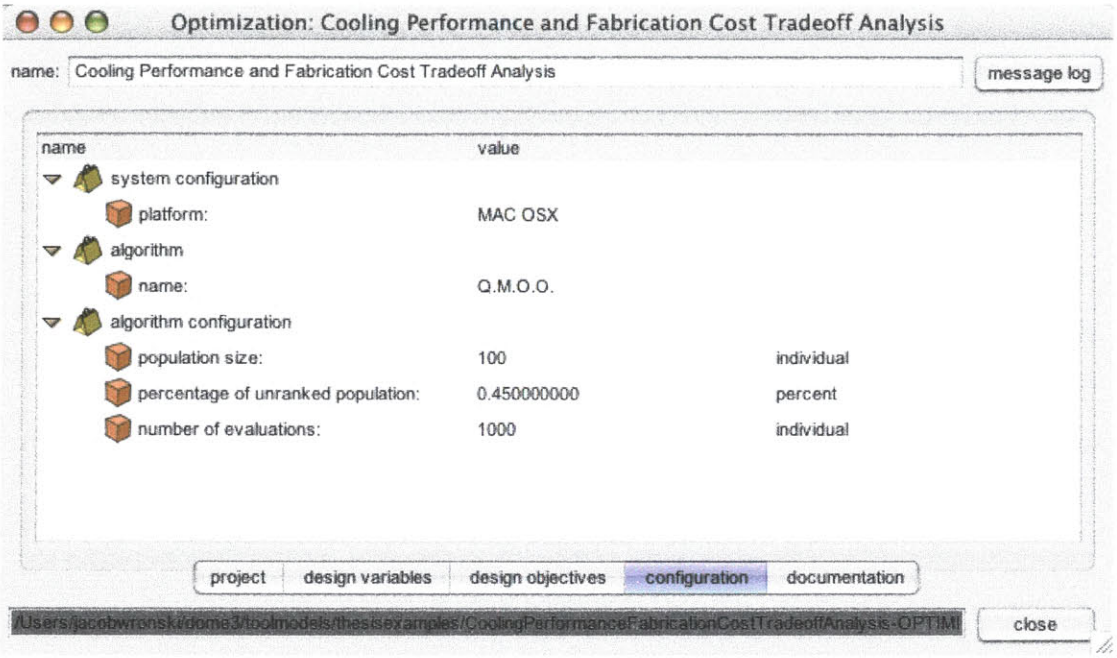


Figure 4-7: Screen shot of the configuration panel for the tube-bundle cooling performance and fabrication cost tradeoff analysis.

- *Platform.* Specifies which platform the model will be published on (ex. Mac OS X©).

- *Algorithm name.* Currently, the optimization tool only supports the *QMOO* algorithm. It is expected that more algorithms will be added in the future.

- *Population size.* The number of individuals to be stored in the algorithms main population at any given time. According to Leyland [46], the optimal population size for QMOO is approximately 90 individuals.

- *Percentage of Unranked Population.* The percentage of population, which will be made up of dominated individuals. This tuning parameter is useful in preserving diversity and exploring multiple local optima.

- *Number of Evaluations.* The number of evaluations to be performed on the objective function before the algorithm terminates itself.

One of the development goals of the optimization tool was to have as few model tuning parameters as possible. Implementing the optimization tool in such manner made it intuitive and easy to use, requiring little knowledge about evolutionary optimization techniques.

Creating Interfaces to Optimization Models

Once published on a DOME server, interface objects allow individuals, with appropriate user permissions, to access and run the underlying optimization services. Multiple interfaces can be created for a single model, and many users can interrogate a single interface concurrently. Also, simulation services can be subscribed to collaboratively, where multiple users interact with the same instance of a DOME optimization model object. Finally, optimization interface objects, like DOME model objects, support custom graphical user interfaces.

The process of creating an optimization tool interface is fairly simple, requiring only that model owner use simple add-and-map techniques to add variable and objective parameter objects to each interface. Each interface has its own configuration panel where the model owner can set use-privileges for the interface before it is published. These privileges include: (i) allowing users to set variable search limits, (ii) allowing users to activate/inactive design variables and objectives, (iii) setting the maximum number of objectives that a service subscriber can run an optimization problem with at any given moment, (iv) and the solution update interval. Figure 4-8 shows a screen shot of an interface object created by the heat transfer engineer in the tube-bundle design scenario.

The description of building optimization models thus complete, the next section will discuss publishing of models on web-accessible DOME servers.

4.3.2 Publishing an Optimization Model on a DOME Server

A comprehensive graphical user interface has been developed to guide users through the process of publishing optimization models on DOME servers. The process is quite trivial and will not be presented here; only the issue of user permissions will be briefly discussed. Model owners deciding to publish their optimization model on a DOME server have complete control over who will be allowed to access the models interface objects, underlying integration project, and resources inside the project. This is to accommodate tight collaboration and secure sharing of intellectual property,

which was identified as a functional requirement stemming from interactions with industrial partners.

With an overview of publishing optimization models complete, the discussion now turns to running a published model to obtain a Pareto-optimal set of designs.

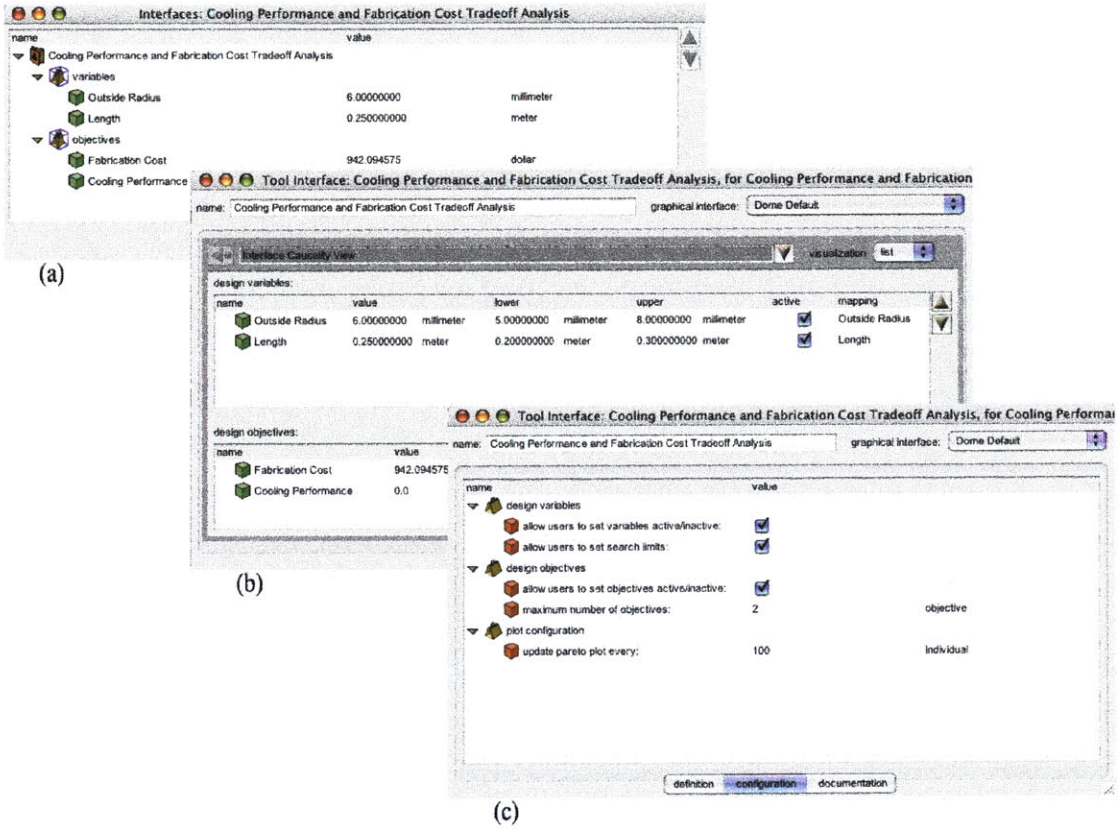


Figure 4-8: (a) Interface manager displays a list of all interface objects belonging to this optimization model. (b) Definition panel groups design variables and design objectives. In this interface, the heat transfer engineer created an optimization problem with two variables (outside radius and length) and two objectives (cooling performance and fabrication cost). (c) Configuration panel allows the heat transfer to set use privileges if other individuals will be allowed to subscribe to the interface object.

4.3.3 Running Optimization Model Simulations

Running optimization model simulations is a simple process. This section will describe the process using the tube-bundle design scenario example. A discussion of the simulation results will follow.

The next section will use the tube-bundle design scenario to demonstrate how the simulation results can be used to set the state of the integrated product model to some selected target design.

Once the optimization model is published on a DOME server, individuals can access the underlying simulation through an interface object via the web. Figure 4-9 shows the heat transfer engineer accessing the optimization model created to evaluate tradeoffs between cooling performance and fabrication cost.

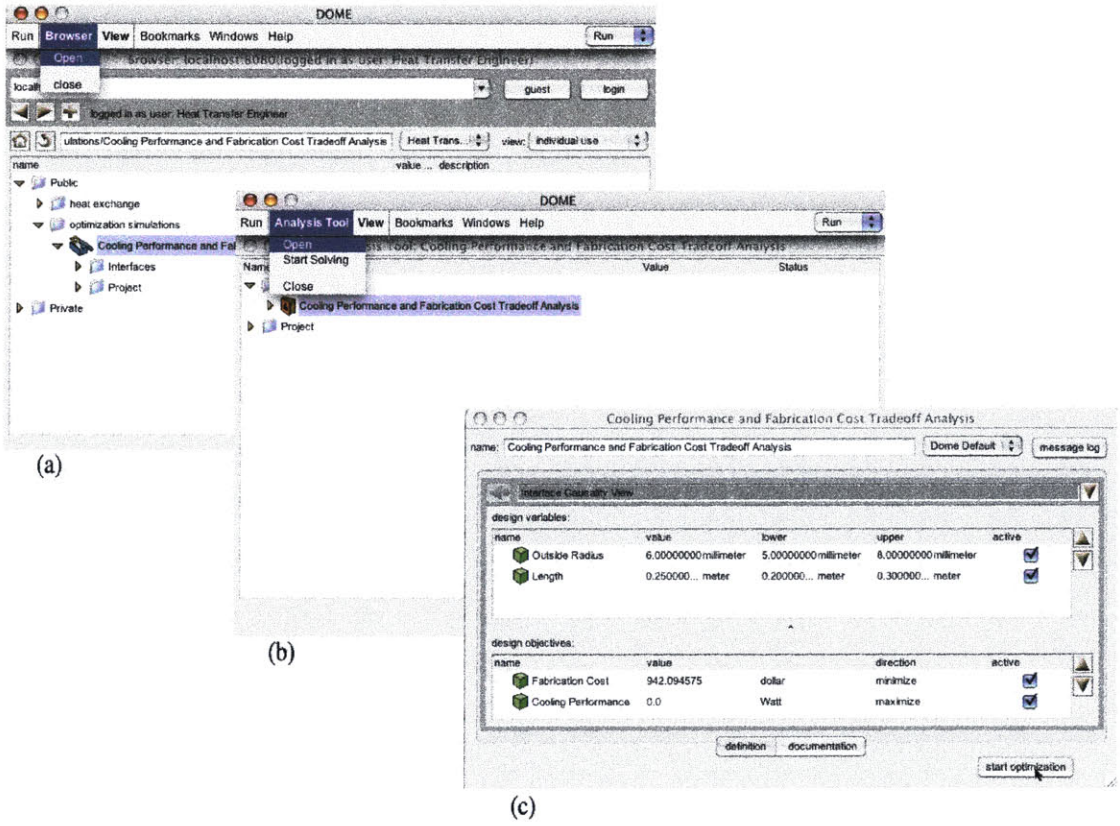


Figure 4-9: (a) The heat transfer engineer has logged into a local DOME server and is now opening the tube-bundle optimization model. (b) The engineer is opening the model's interface object to run the underlying optimization simulation. (c) Optimization model interface object.

To run the model simulation, an individual must simply operate the graphical user interface. The optimization engine, discussed in Chapter 3, controls the execution of the model simulation throughout this process. Initially, it will configure the optimization problem with the tuning parameters spec-

ified by the model builder. Two important tuning parameters are population size and number of evaluations. Next, the optimization engine will create the initial population and will randomly assign it design variable values that fall within the specified upper and lower limits. Each individual will then be evaluated and the resulting design objective values will be assigned to it. If the individuals are found to be lacking, they are removed from the population, otherwise they join the main population to become parents of new individuals (see Figure 3-1). Crossover and mutation operators, discussed in §3.4, are used to assign design variable values to new offspring. This process is repeated until the termination criterion is reached, which is specified by the model builder via a tuning parameter.

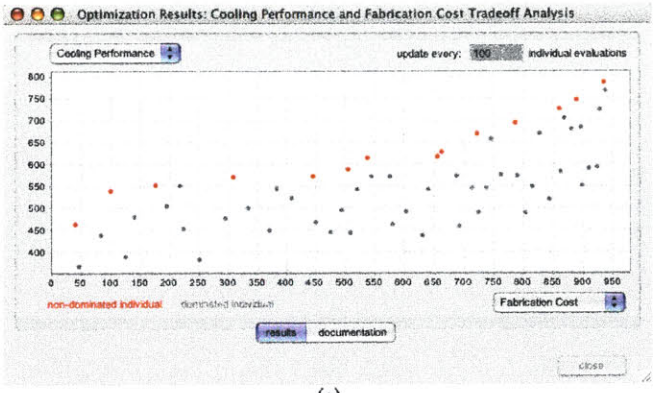
While the algorithm is executing the model simulation, the interface counts the number of times that the underlying integration project has been evaluated. When some user specified number of evaluations has been reached, the optimization engine will output the current state of the population to the interface for the model subscriber to review. The mechanisms responsible for this will be discussed in detail in §4.4. Figure 4-10 shows the heat transfer engineer receiving results from the optimization tradeoff analysis. The population is displayed in two instances, at 100 evaluations, and at 1000 evaluations where a significant improvement in the convergence of the population to the POF can be observed. This result was obtained with a population of 100 individuals.

Although the optimization tool developed in this work supports optimization of models with more than two design objectives, the results visualization method does not. Current development efforts include implementation of surface visualization techniques for more than two design objectives.

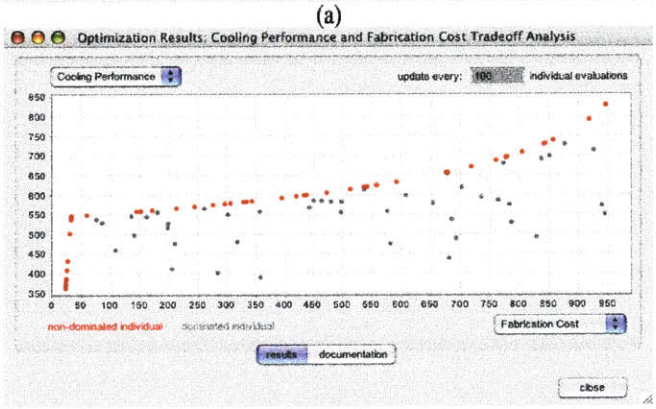
4.3.4 Evaluating Design Tradeoffs

The result of running an optimization simulation is a characterization of all the interesting regions of the integrated product model. Individuals running optimization services are presented with a family of Pareto-optimal design candidates, where each candidate represents different performance in each of the design objectives under consideration. Design candidates are Pareto-optimal when they do not dominate each other (i.e. a selected individual does not perform better than any of the other individuals in the set in every design objective).

Once presented with the results, the decision maker is empowered to explore all of the interesting



(a)



(b)

Figure 4-10: (a) The tube-bundle optimization problem after 100 evaluations, showing individuals dispersed around the design objective space. (b) The same optimization problem after 1000 evaluations. Individuals have converged to the Pareto-Optimal frontier. From designs with best cooling performance, but high fabrication cost (top right) to designs with low fabrication cost, but poor cooling performance (bottom left). Red dots indicate non-dominated individuals (or optimal solutions) in both graphs. Fabrication cost is calculated per 1000 tubes.

regions of a model, gain insight into the systems behaviour, evaluate tradeoffs between the presented designs and select a favourable design candidate to derive the final state of the product system model.

In the tube-bundle design scenario, the heat transfer engineer is presented with a population of design candidates ranging from designs with excellent cooling performance, but high fabrication cost, to designs with low fabrication cost, but poor cooling performance (see Figure 4-10). Immediately, a number of interesting observations can be made about the behaviour of the system. First, there is quasi-linear region of the model where the sensitivity of cooling performance to changes in fabrication cost is low. For example, a fabrication cost increase of 1,500% ($0.05/tube to $0.80/tube)

corresponds to a cooling performance increase of only 36% (550W to 750W). Secondly, for a tube with a fabrication cost below \$0.05, there is a sharp decrease in its cooling performance. Finally, there is a region in the model (fabrication cost >\$0.80) where the cooling performance appears to be increasing quadratically with increasing fabrication cost. The engineer considers tradeoffs between different designs and selects a particular design point on the POF to be the initial starting design for the tube. The design variables corresponding to the selected design targets are then passed into the integration project so that a complete, and parametrically consistent, system state can be derived. Figure 4-11 provides a screen image of this process.

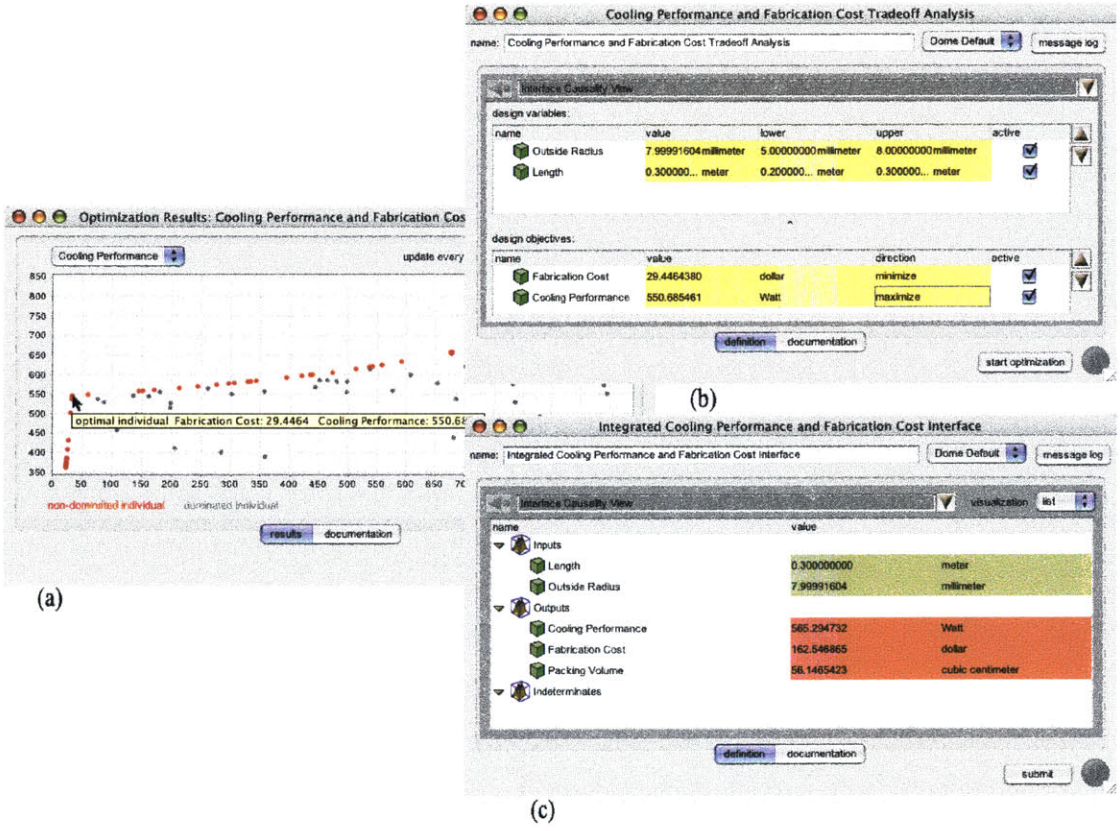


Figure 4-11: (a) The heat transfer engineer selects target values for the two design objectives. (b) Target values of the design objectives and corresponding design variables (inputs to the integration project) are passed to the optimization model interface. (c) The input design variables that correspond to the target design objectives are also passed to the integration project interface to derive a new, parametrically-consistent state of the system model.

This use of the optimization tool in the tube-bundle design scenario is an example of a target-driven design approach. The heat transfer engineer used the optimization tool to elucidate and evaluate

system model tradeoffs and then selected a favourable set of design objectives (i.e. design targets) as a starting point. Selected design targets were then passed to the integration project interface to derive all other published information about the integrated product model. An alternative, trial-and-error approach, would have the engineer changing the values of the inputs to the simulation until a favourable set of design objectives was revealed through the calculation. Both approaches to the engineering design process will be further examined in §4.5.

4.4 Flow of Information Between QMOO and DOME

Communication between QMOO, the core of the optimization engine, and an integrated product model in DOME is illustrated in Figure 4-12. During an optimization simulation, information is passed from the optimization algorithm, to the integration project and back (Figure 4-12a). Since QMOO has no knowledge about the solving mechanisms of DOME, a custom DOME evaluator object had to be written to make communication between QMOO and DOME possible. A second communication pathway, between QMOO and a client interface object, is shown in Figure 4-12b. The DOME monitor object takes a snapshot of the population at some user-specified, number of individual evaluations, and sends that data through the interface to the client. Both, the DOME evaluator and monitor objects were developed in this work will now be discussed in detail.

A third object, also created in this work, is responsible for tuning the optimization algorithm. Its primary role is to translate information entered in the model configuration panel, discussed in §4.3.1. Tuning parameters include: number of evaluations, population size and percentage of dominated individuals in the main population. This configuration object could also allow the model builder to select different mutation and crossover operators, clustering methods, and ranking techniques. However, the author of this work felt that exposing too many algorithm-tuning parameters would make the optimization tool difficult to use. Furthermore, modifying these tuning parameters during the development of this tool had little effect on the performance of the algorithm.

4.4.1 The DOME Evaluator Object

The evaluator object allows QMOO to communicate with a DOME-native optimization model and carries out two important functions. First, it reads the DOME-native optimization model and extracts information required by QMOO to completely define the optimization problem. This informa-

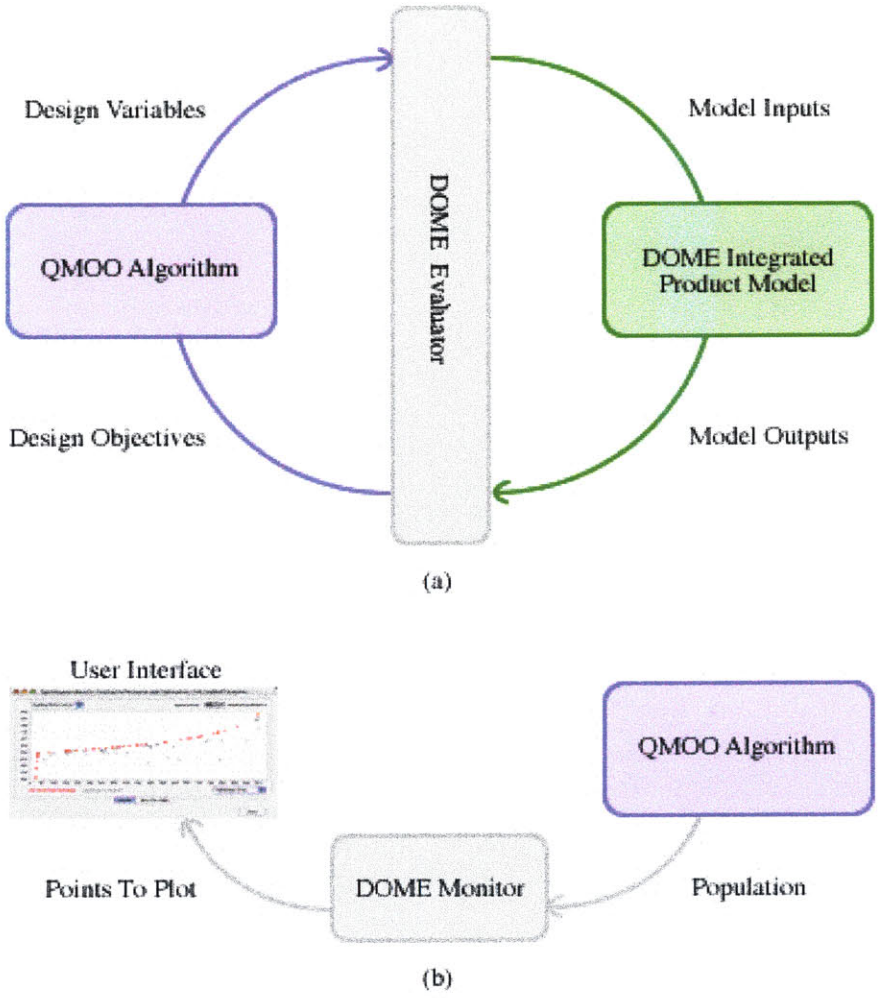


Figure 4-12: (a) A 'DOME Evaluator' object serves as a communication bridge between QMOO and DOME. Variables are generated by QMOO and passed into the DOME integration project as inputs. The project is solved and the calculated results are returned to QMOO as design objectives via the 'DOME Evaluator'. (b) At some interval of evaluations, QMOO will pass a 'snap-shot' of the entire population to a 'DOME Monitor' object. The 'DOME Monitor' notifies the client interface object that a population is ready for viewing, which the client interface then displays.

tion includes the number, and data type, of all decision variables and design objectives. The second function of the evaluator object is to manage the flow of data values throughout the execution of an optimization problem.

When an individual is being evaluated, the evaluator object must match each of its design variables with a corresponding input to the integration project inside the optimization model. Design variable values are then copied to the input parameters of the integration project and the model starts solving. When the model solving is finished and the project outputs are ready, a map inside the evaluator object is used to find an optimization model design objective for each corresponding integration project output. The parameter value of the integration project output is then copied to the design objective parameter and modified, depending on whether the objective is to be maximized or minimized. If no outputs are returned by the project, likely since one or more of the models inside it have crashed, the individual is marked as unfeasible and removed from the population. In both cases, QMOO is notified that the evaluation of the individual is complete, and the individual is queued for subsequent processes or removed from the population.

The pseudo code of the DOME evaluator object is included in Appendix A.

4.4.2 The DOME Monitor Object

As mentioned before, the DOME monitor object is charged with the task of sending the current population of individuals in an optimization problem, to a client interface object for visualization. The interface user is then able to interact with the published optimization model.

The working mechanism of the monitor object is trivial and requires only one tuning parameter, which is the client-side population update interval. During the execution of an optimization, the number of individuals evaluated for their design objective values is counted. When this number reaches the update interval value, the monitor object is called and its main process is executed in two parts. First, the client side visualization plots are reset so that the new version of the population may be plotted. Next, the current state of each living individual in the population is plotted. This includes sorting the individuals into dominated and non-dominated sets. After the monitor object has finished plotting each member of the population, it waits until the next update cycle.

The pseudo code of the DOME monitor object is included in Appendix A.

4.5 Implications of Target Driven Design for Product Development

The optimization tool developed in this work provides an alternative approach to the traditional engineering design process. Specifically, it addresses two practical design challenges at the concept development stage: (i) integrated assessment, and (ii) tradeoff driven design process. Integrated assessment allows product development teams to more effectively explore the design space of a product, run a higher number of "what-if" scenarios, and identify a promising design candidate early in the process. In tradeoff driven design, teams can specific a set of optimal tradeoffs and then see what configurations yields them, rather than guessing a design configuration using a 'trial-and-error' approach and observing its effect on the design objectives. These implications will now be discussed in detail.

First, the size and scope of most product development projects severely prohibit a thorough exploration of all potential design candidates and the selection of a promising starting design. Ulrich and Eppinger [72] state that the goal of concept generation is to thoroughly explore the space of product concepts that may address the customer needs. Wallace et al. [77] observed, through interactions with industrial research partners, that evaluation of a single design candidate took months to complete, resulting in only 20 design cycles occurring on average. Using the DOME simulation environment, the same design cycles required only 20 seconds to 1 day, depending on the type of changes and quality of analysis.

The optimization tool allows individuals subscribing to optimization model services to run many more improvement cycles on the product system model while still reducing product development time. Furthermore, the use of heuristic evolutionary algorithms ensures that exploration of the design space is directed. Each new potential design candidate will perform better in the specified design objectives than its predecessors. Combining the seamless integration capabilities of DOME with the directed search techniques of evolutionary algorithms empowers designers to effectively explore the design space of a product.

The second design challenge is the issue of design tradeoffs in product development projects. Ulrich

and Eppinger [72] state that one of the most difficult aspects of product development is recognizing, understanding, and managing such tradeoffs in a way that maximizes the success of the product. This is especially true in multi-disciplinary product development teams comprised of individuals in different geographical locations with unique areas of expertise. In such an environment, obtaining an integrated system view is difficult and decisions about global tradeoffs are often left to intuition [63].

The optimization tool implements a multi-objective optimizer that facilitates the rapid elucidation and evaluation of design tradeoffs. Using Pareto optimality, the optimization tool is able to generate the optimal tradeoffs curve. This curve represents a family of points (or designs) that represents optimal tradeoffs between the design objectives under evaluation. Moving from one design point on the curve to any other point will make the performance of at least one design objective worse. Having this set of Pareto-optimal designs, the decision maker can then select the optimal tradeoff configuration to derive the design of the entire product system (see §4.3.4).

For the reasons discussed above, the design tool developed in this work, offers product development organizations an alternative approach to concept development. Decision makers are empowered to: (i) create a digital product system model, (ii) explore a high number of potential design candidates, (iii) evaluate optimal design tradeoffs between a few best candidates, and (iv) select a starting target design that best performs under the specified design objectives. Once a complete concept design is obtained, the product development team can proceed to subsequent prototype building and testing stages.

Table 4.2 provides a comparison between the proposed approach to concept development and the traditional trial-and-error approach presented in Chapter 2.

4.6 Summary

The optimization tool presented in this chapter was developed for the DOME modeling environment to facilitate the rapid elucidation and evaluation of design tradeoffs in a product development project. Individuals can use this design tool to identify design candidates that perform extremely well in terms of the modeled design objectives, early in the design process.

Stage	Traditional Process	Target-Driven Process
Generation of concepts.	•Difficult to obtain integrated system view, results in only a few design concepts being generated.	•DOME-enabled system integration allows design cycles to take minutes, as opposed to days, to be evaluated. •As a result a large number of design candidates can be generated.
Selection of concepts.	•Design candidates are generated in an ad-hoc fashion with a 'trial-and-error' approach. •Local design experts make intuitive guesses about the behaviour of the system and select (what appears to be) the best design candidate.	•Evolutionary algorithms provide a smart design space exploration technique. •A set of optimal design candidates is presented to the decision maker. •Global tradeoffs are presented to the decision maker on Pareto-optimal frontiers.
Testing of conepts.	•Design experts spend considerable time coordinating information to evaluate a selected design concept against any number of (potentially multi-disciplinary) design objectives.	•Designers are empowered to select 'design targets' from the computed optimal design tradeoffs. •Selections are then used to quickly synchronized the complete product system model to derive the selected design.

Table 4.2: Contrast between the traditional engineering approach to concept development, and the optimization tool-enabled approach. Concept development stages adopted from Ulrich and Eppinger [72].

To help clarify the formal description of the optimization tool, a trivial design scenario was introduced. It involved an in-house heat transfer engineer collaborating with an external part supplier to derive an optimal tube geometry design while balancing two competing design objectives: cooling performance and fabrication cost. The heat transfer engineer used a DOME-native model object to model the cooling performance of a tube. The external part supplier used a third-party application, EXCEL™, to provide the fabrication cost services. Both experts published their services on web-accessible DOME servers.

Four aspects of the optimization tool were illustrated using the tube bundle design scenario. They are: (i) building an optimization simulation, (ii) publishing that simulation service on a web-accessible DOME server, (iii) running a published model to obtain a Pareto-optimal set of solutions, and (iv) evaluating optimal design tradeoffs and selecting target designs for subsequent design processes. The notion of selecting design targets to derive a particular system design is being referred to as the target-driven design approach.

To make information transfer between the optimization engine (QMOO) and DOME compatible, a number of custom objects had to be implemented. The DOME evaluator object is responsible for managing the flow of parameter data between the optimization algorithm and the DOME integration project during the execution of a simulation. The DOME monitor object is responsible for displaying the results of an optimization simulation to a remote or local interface client. Finally, a configuration object was implemented and its role is to simply tune the optimization algorithm (ex. population size, number of evaluations before termination).

The optimization tool developed in this work has significant implications on the traditional engineering design approach. Seamless integration of models and services allows product development organizations to explore a large number of potential candidates in the early concept development stage. The use of heuristic optimization techniques ensures that the search is directed and that each newly discovered design candidate is better than its predecessors. Another benefit of this design tool is its ability to explore design tradeoffs between the specified design objectives. Decision makers can readily evaluate an entire population of optimal performance tradeoffs on a Pareto-optimal curve, select a promising candidate, and derive the complete system model design.

The introduction of the optimization tool and the target-driven design approach it facilitates is now complete. The next chapter describes a validation study of the target-driven design approach.

Chapter 5

Application

5.1 Introduction

To validate the *target-driven design* approach, a real-world engineering design scenario will now be presented. It is a multi-objective design problem, of a hybrid energy system for delivering electricity using a diesel engine generator and a photo-voltaic (PV) array. The design models that constitute this optimization problem were originally developed by Sukkasi [70] as part of an *Alternative Energy Design Toolkit* created for the modeling environment, DOME, discussed in §2.3.2.

In the original work, this multi-objective design problem was "optimized" using the following methodology. First, the problem was decomposed into an equivalent number of single-objective optimization problems – in this case five. Next, the performance of each design objective was analyzed using crude response surface techniques[1], where the design objective was plotted as a function of the two decision variables at regular intervals throughout the design space. Finally, a design was chosen, which performed best under the most critical design objective and any subsequent tradeoffs between competing design objectives were resolved by considering how well the proposed design meets the original performance goals.

While the approach outlined above is effective in the design and optimization of a hybrid PV-diesel engine system, it has two important limitations when extended to most engineering applications. First, information about the performance of each design objective was obtained using an enumera-

[1]For a detailed description of response surface methodologies, the reader is referred to Myers and Montgomery [50].

tive search technique. Although this "trial-and-error" technique worked well in this design scenario, its use in most practical engineering problems is limited for reasons discussed in §2.4.5. Secondly, design tradeoffs were resolved by manual analysis of individual objective function surface plots, without any tools for elucidating such tradeoffs between the objectives. This method may prove to be difficult for more complicated optimization problems with many design objectives.

The optimization tool developed in this work will be used on the hybrid energy-system design scenario to illustrate how it facilitates the effective elucidation and evaluation of design tradeoffs in multi-objective design problems. Although this design problem has been first introduced in [70], its optimization using a multi-objective optimizer, and the discussion of the results in the context of target-driven design, are original to this work.

5.2 Design of a Hybrid PV-Diesel Energy System

The discussion of the design scenario presented in this section will begin with background and problem description. Next, the constituent design models used in the scenario will be briefly mentioned, followed by a discussion of their integration architecture with the optimization engine. Subsequently, the results of the optimization problem will be presented and discussed. Finally, the present design scenario will be used to illustrate how it empowers designers to practice a target-driven, as opposed to a "trial-and-error", design approach.

5.2.1 Background

In remote villages far from established utility grids, electric energy is often supplied by diesel engine generators or local hydroelectric plants. However, diesel engine generators cannot be considered as a long-term solution because of CO_2 emissions, the price inflation of fuel, and the increasing penalty cost due to environmental protection policies adopted by many countries [8, 61]. Alternatively, PV arrays offer an environmentally-friendly method of supplying electric energy. Unlike in the case of diesel engines, the initial energy invested in the production of a PV array is recovered completely during its operation -- when the PV array generates electricity without emitting any pollutants[2]. This makes PV arrays the preferred technology for supplying energy. However, they are more ex-

[2]The length of time it takes to completely recover all of the energy invested during the production phase of a PV array is known as the *energy payback period* [47].

pensive to produce than diesel engine generators, which has thus far prohibited their widespread use.

Increasing diesel fuel prices along with the high cost of transporting fuel to remote locations are making hybrid PV-diesel generation systems competitive with diesel-only generation [18]. In a hybrid PV-diesel energy system, a PV array is used in addition to a diesel engine to generate electricity. To a designer of such a system, the PV array is an expensive method of providing environmentally-friendly electricity, while the diesel engine provides inexpensive electricity, but is taxing on the environment. Therefore, to design this system one must negotiate tradeoffs between two competing design objectives – environmental impact versus cost of the hybrid system.

5.2.2 Design Scenario Description

This design scenario is assumed to take place on a remote island in southern Thailand with a system operation period of 20 years [70]. The amount of electricity generated by each component corresponds to the specified PV load fraction – a design parameter, ranging from 0 to 1, which specifies how much of the total load is to be delivered by the PV. A second design parameter, number of diesel operating hours, specifies the number of hours the diesel engine is to operate each day. The design objectives are the system's capital cost, life-cycle cost, net electricity cost, CO_2 emission, and electricity production efficiency.

The hybrid energy system will be designed for two different applications: (i) for generating electricity at a private island resort, and (ii) for providing affordable electricity to the island's inhabitants. Each application has its own set of prioritized design objectives, which will be discussed in detail in §5.2.6.

5.2.3 Constituent Design Models

The design models used in this design scenario came out of the *Alternative Energy Design Toolkit* developed by Sukkasi [70]. They are:

- PV system load breakdown
- PV system life-cycle costing
- Engine-generator system life-cycle costing

- Simplified PV module and battery energy analysis
- Simplified diesel generator energy analysis
- CO_2 emissions from electricity generating systems

The critical input values to these design models are shown in Table 5.1. For a detailed discussion of the individual design models, the complete set of input values and how these values were decided upon, the reader is referred to [70].

Design Model	**Input Parameter Description**	**Parameter Value**
PV load breakdown	net daily load	4071 Wh
	day time load	37% of net load
	sufficient solar irradiance	13.5 h/day
	battery charge efficiency	0.8
	battery discharge efficiency	0.95
	minimum allowed state-of-charge	0.7
Engine-generator / PV life-cycle costing	discount rate	0.1
	excess inflation rate	0.0
PV energy	PV array lifetime	20 yr
	battery lifetime	5 yr
Diesel energy	3000 RPM engine lifetime	6000 h
	1500 RPM engine lifetime	10000 h
	diesel fuel heating value	10.08 kWh/l
CO_2 emission	fuel consumption	2.45 kg/l
	manufacturing of PV array	0.267 t/kW-yr
	manufacturing of battery	0.062 t/kW-yr
	manufacturing of diesel engine	0.069 t/kW-yr

Table 5.1: Input values to the individual models in the hybrid PV-diesel energy system design scenario, taken from [70].

Integration of the individuals design models is illustrated in Figure 5-1. Inputs to the system model are the PV load fraction and the number of diesel engine operation hours. They are used by the DOME integration model to calculate all required input parameters to the models inside the integration – namely, the PV rated and operating power, and the diesel engine rated and operating power.

The outputs of the integration are the total life-cycle cost, net electricity cost, total CO_2 emissions, net electricity production efficiency and the total capital cost.

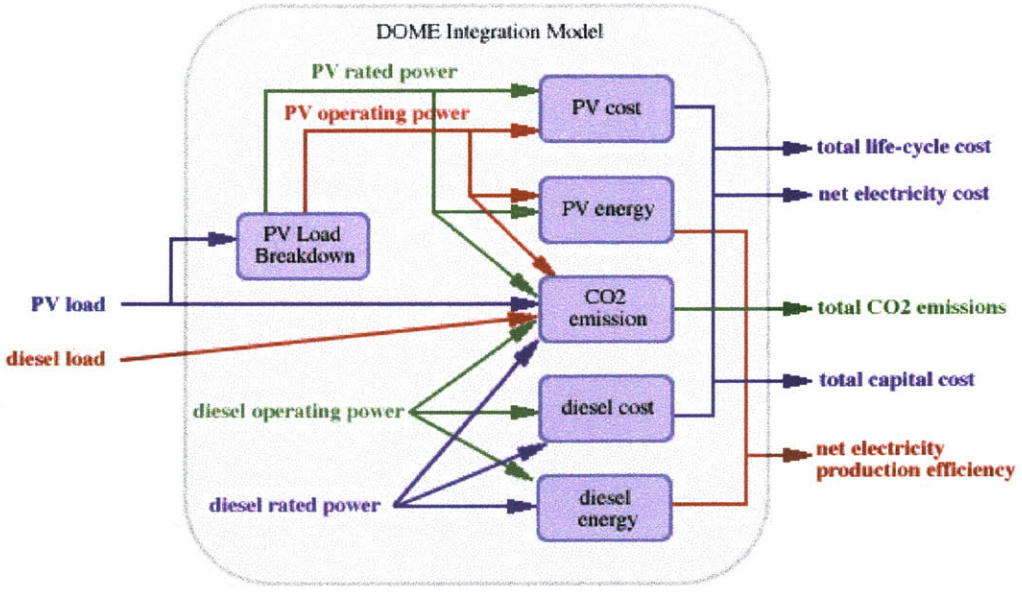


Figure 5-1: Integration of the individual simulation models in the hybrid PV-diesel energy system design scenario. Connections are made declaratively and the directions of information flow are determined by solving mechanisms within DOME. (Reproduced from Sukkasi [70], pg. 110)

5.2.4 Optimization of the Design Scenario

The hybrid PV-diesel energy system design scenario is a multi-objective optimization problem with two independent design variables and five design objectives. The two design variables and their lower/upper limits are shown in Table 5.2. Initially, a range of 0 to 1 was used for the PV load fraction. However, this lead to the optimization problem crashing whenever the algorithm evaluated the design objectives at either extreme – due to a division by zero in the PV power calculations. For similar reasons, a valid range had to be selected for the second design variable.

The five design objectives are shown in Table 5.3, along with whether a given design objective is

Design Variable	Limits
PV load fraction	0.05 to 0.95
daily hours of diesel operation	1 to 24

Table 5.2: Independent design variables and their lower/upper limits.

to be minimized or maximized. Some of the design objectives will compete against each other – forcing the system designer to resolve tradeoffs between them. For example, the total CO_2 emissions can only be minimized if more of the load is supplied by the PV system, however this will result in a design with a higher net electricity cost and total capital cost. As mentioned earlier, in practice such design tradeoffs are difficult to elucidate and can only be resolved after careful consideration of the original application for which the system was intended. The design tradeoffs of this system will be discussed in section §5.2.5.

Design Objective	Direction of Optimization
total life-cycle cost	minimized
net electricity cost	minimized
total CO_2 emission	minimized
net electricity production efficiency	maximized
total capital cost	minimized

Table 5.3: Design objectives and how each is to be optimized.

This multi-objective problem was solved using the optimization tool developed in this work. A population of one-hundred individuals was used and up to 45% of the population's individuals were dominated – this was done to help preserve population diversity and to avoid early convergence to a local optimum. Finally, the algorithm was terminated after one thousand individual evaluations. The configuration panel of the user interface to this optimization problem is shown in Figure 5-2. Each individual evaluation took approximately five seconds[3], for a total simulation time of eighty minutes. The results of the optimization will be presented and discussed in the context of target-driven design in the following section.

[3]simulation was executed on an Apple PowerBook™with a 1GHz PowerPC G4, 1MB L3 cache processor and 512 MB of SDRAM memory

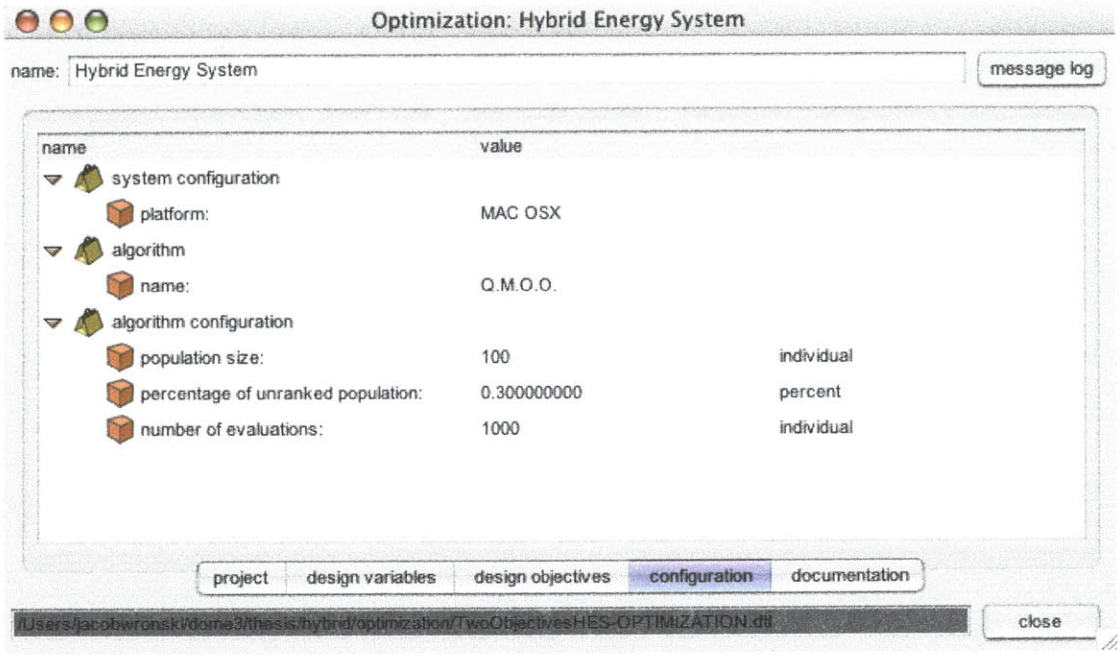


Figure 5-2: Configuration panel of the hybrid PV-diesel energy system optimization. Key algorithm tuning parameters are the population size (100), percent of unranked individuals (30%) and total number of evaluations (1000).

5.2.5 Results

The interface used to interact with the underlying optimization simulation is shown in Figure 5-3. To run the simulation, individuals follow the procedure outlined in §4.3.3. Multiple simulations can be executed concurrently and users decide which design objectives to evaluate and which to deactivate. Figure 5-4, Figure 5-5 and Figure 5-6 show two-dimensional Pareto curves, which illustrate optimal tradeoffs between the design objectives. Figure 5-4 shows the total electricity cost increasing with decreasing CO_2 emissions. At the leading edge, a hybrid engine system can be designed with a reduction of 33% in total CO_2 emissions, at a cost increase of only 2%. Reductions in CO_2 continue to decrease as cost of the system increases. Finally, at the bleeding edge, the cost of the energy system increases 54% with only a 4% reduction in total CO_2 emissions. Figure 5-5 dictates that an increase in the net electricity production efficiency can only be achieved with an increase in the net electricity cost – a more efficient hybrid energy system is more expensive to fabricate. Possible designs range from an energy system with an electricity production efficiency and electricity cost of 38% and 0.22 $/kW-h, respectively – to an energy system with an efficiency and cost

of 69% and 0.36 $/kW-h. Figure 5-6 graphs the relationship between the net electricity production efficiency and total CO_2 emissions. For most of the graph, the results are intuitive – a more efficient energy system is expected to convert more of its input energy into useful output energy and should therefore produce less CO_2 emissions. However, as the insert in Figure 5-6(b) suggests, at efficiencies higher than 65%, the two design objectives begin to compete.

The population in each simulation has been plotted in its decision variable space and the graphs are included in §B. For the minimization of the net electricity cost and total CO_2 emission, there are two clusters of individuals, where each cluster represents designs with best performance in only one of the specified design objectives. For example, designs located at a PV daily load fraction of 0.05, all provide electricity at a low cost, but they also emit the highest amounts of CO_2. Individuals located between the two clusters, represent designs with optimal performance tradeoffs between the two design objectives. The second optimization problem, namely the minimization of electricity cost and maximization of electricity production efficiency, showed similar behaviour of the population in the decision variable space. Interesting behaviour of the population can be noted for the final optimization simulation – maximization of the net electricity production efficiency and minimization of the total CO_2 emission. The population after 100 evaluations is scattered randomly throughout the design space, but after 1000 evaluations all non-dominated (optimal) individuals converge on a straight line where the PV load fraction is equal to 95%. This indicates that the two design objectives in this problem have an inverse relationship for only one of the design variables – the daily diesel operating time. Furthermore, this result was expected as it is known that PV arrays are more efficient at generating electricity and produce less CO_2 emissions than diesel engines. Increasing the daily diesel operating time reduces the total CO_2 emissions, but it also has the disadvantage of reducing the electricity production efficiency. This is because the same amount of electricity can now be delivered over a longer time period, which can be accomplished with a smaller, less polluting, diesel engine. Lastly, the cluster of optimal individuals covering the vertical line (PV load fraction = 0.95), represents optimal tradeoffs between the two design objectives for different values of the daily diesel operating time.

The simulation results were used to design a hybrid energy system for the applications discussed in §5.2.2. These design examples validate the target-driven design methodology and will now be discussed.

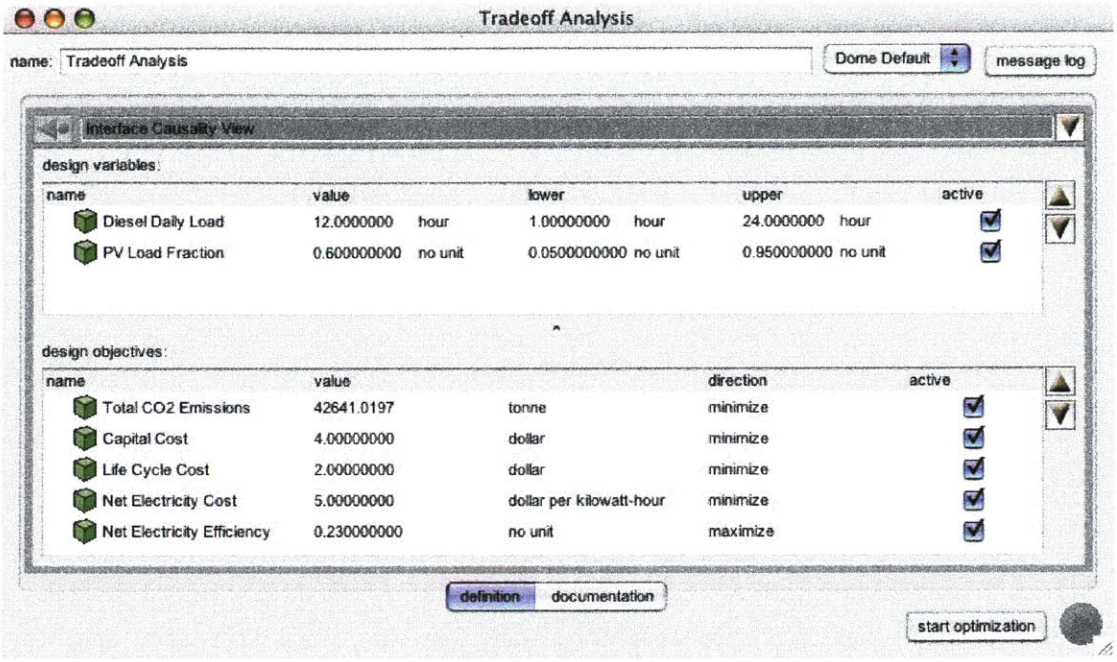
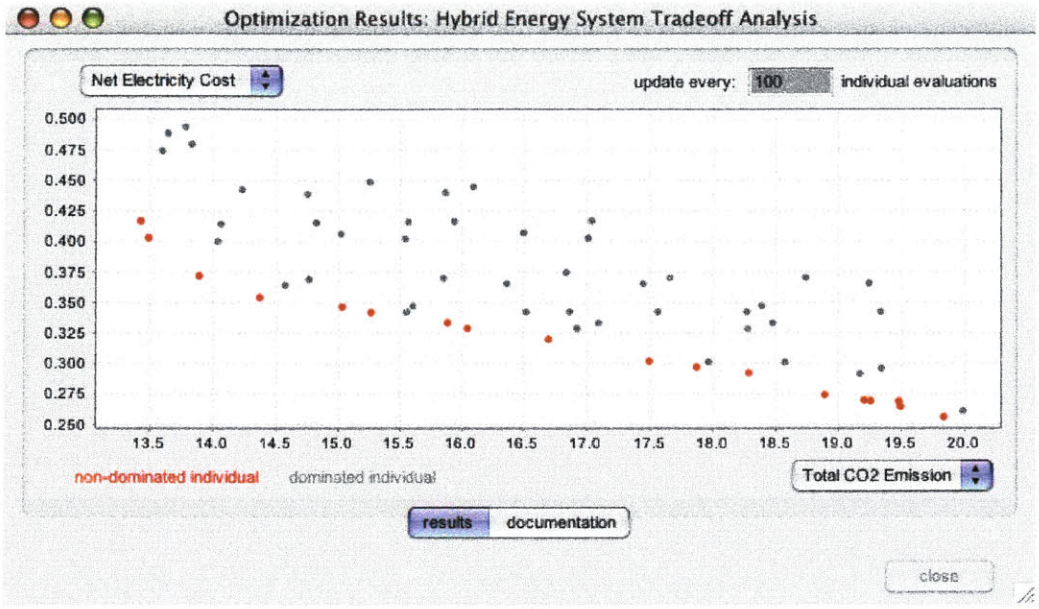


Figure 5-3: Graphical interface to the hybrid PV-diesel energy system optimization simulation.

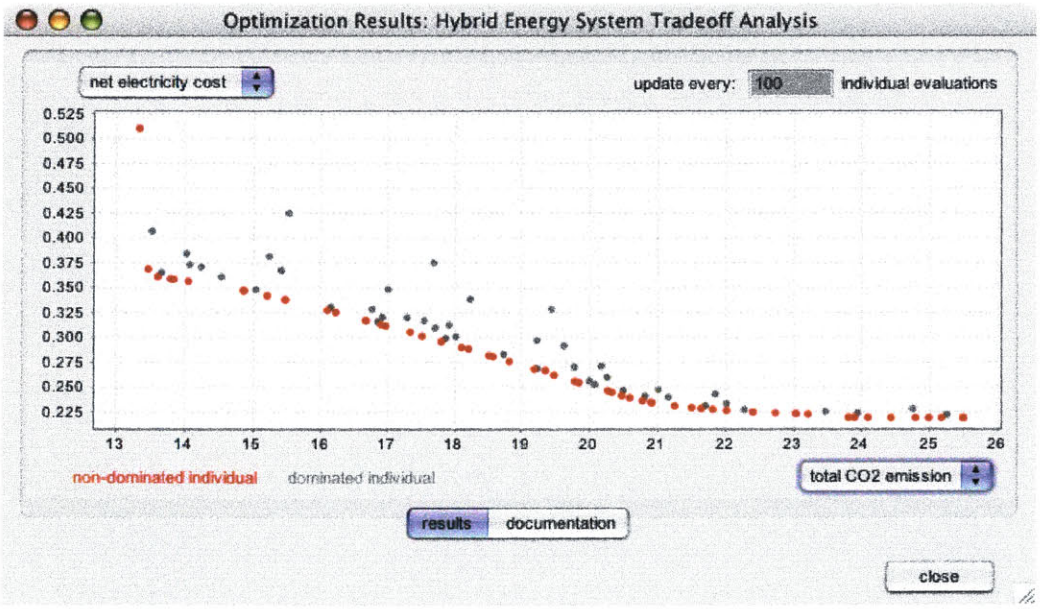
5.2.6 Target-Driven Design Examples

Individuals can use the simulation in Figure 5-3, to design a hybrid energy system, which best meets the performance goals of their particular application. The underlying optimization tool empowers designer to quickly resolve tradeoffs between competing design objectives, as illustrated with examples in §5.2.5. In addition, system designers can seamlessly send performance targets for the design objectives of interest, into the system model, to derive a design that will achieve these targets. This empowers individuals to deviate from the traditional "trial-and-error" design approach to a "target-driven" one.

This concept will now be illustrated with the design of a hybrid energy system for two applications: (i) a private island resort, and (ii) providing affordable electricity to the inhabitants of an island. In each case, the principle designers interact with a third-party contractor providing the simulation service in Figure 5-3. However, each principal designer has a unique set of performance targets, which must be satisfied in order to make the hybrid energy system successful in their respective

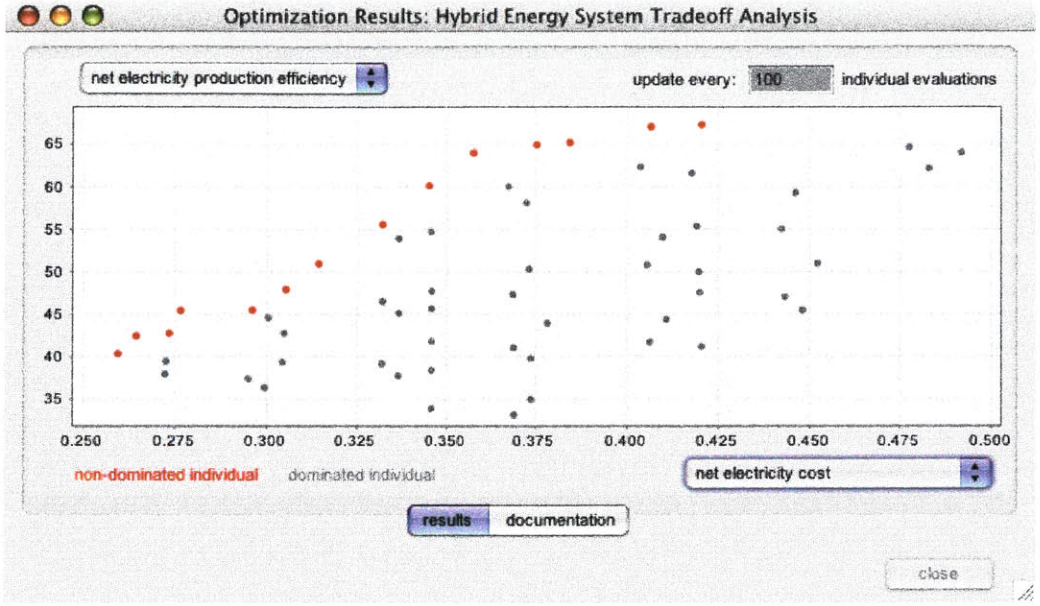


(a)

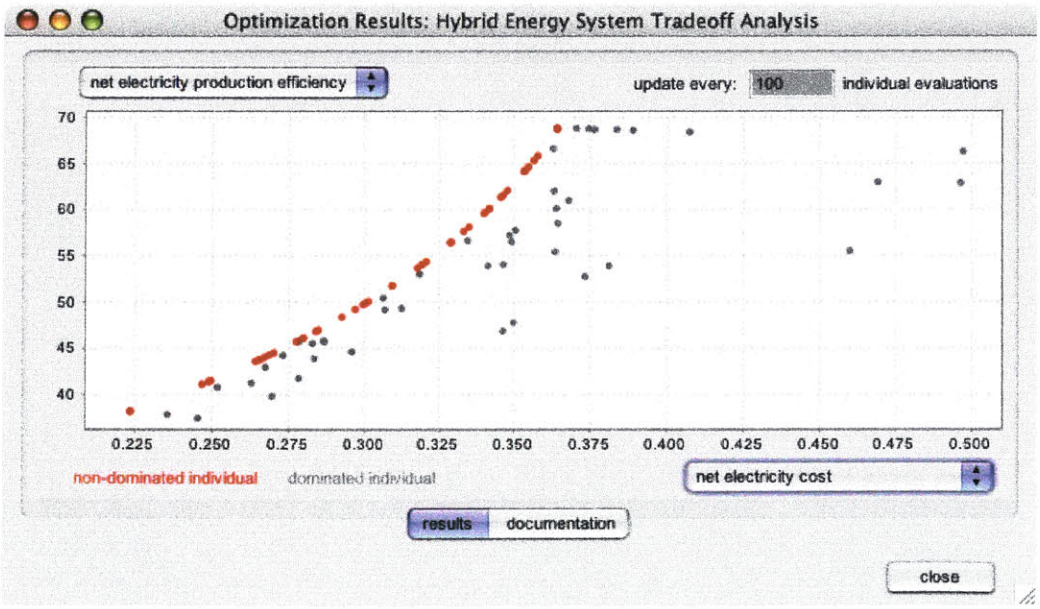


(b)

Figure 5-4: Optimization results of the hybrid energy system design scenario for two competing design objectives: *net electricity cost* vs. *total CO_2 emission*: (a) population after 100 individual evaluations, and (b) final population after 1000 individual evaluations. Final population of 76.

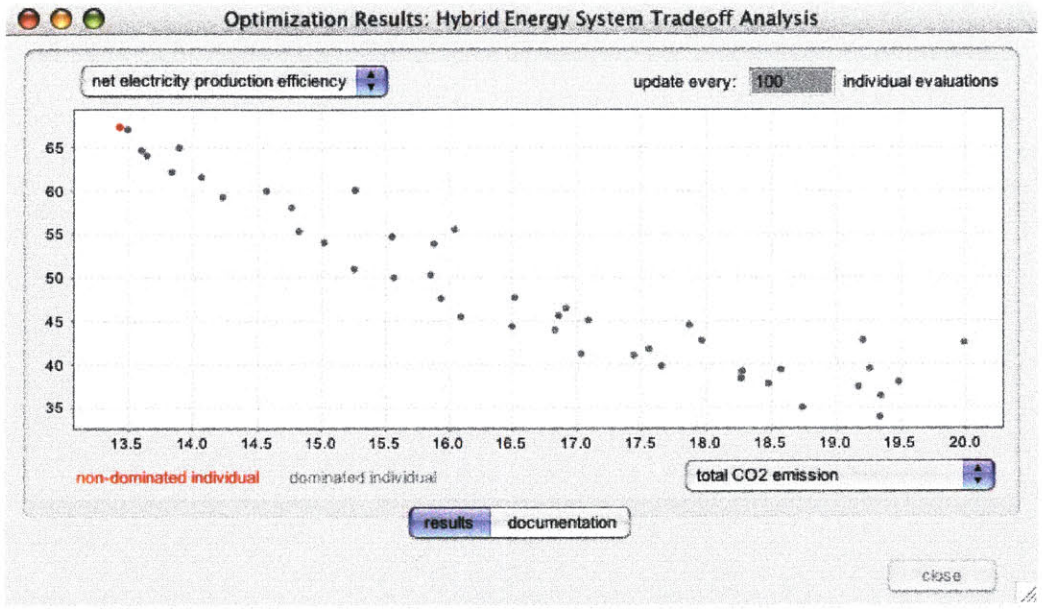


(a)

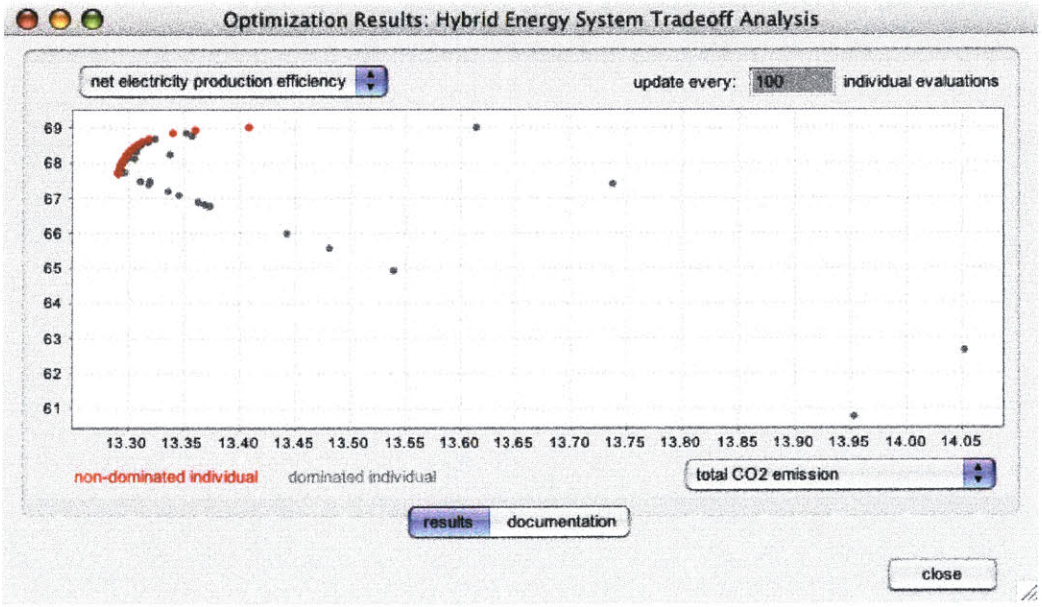


(b)

Figure 5-5: Optimization results of the hybrid energy system design scenario for two competing design objectives: *net electricity production efficiency* vs. *net electricity cost*: (a) population after 100 individual evaluations, and (b) final population after 1000 individual evaluations. Final population size of 107.



(a)



(b)

Figure 5-6: Optimization results of the hybrid energy system design scenario for two competing design objectives: *net electricity production efficiency* vs. *total CO_2 emission*: (a) population after 100 individual evaluations, and (b) final population after 1000 individual evaluations. Final population size of 77.

application. Figure 5-7 provides a summary of these interactions.

Generating Electricity at a Private Island Resort

Private island resorts depend on a clean environment to attract tourism. In such an application, it is reasonable to make the assumption that a successful hybrid energy system would have to provide clean energy above all. In terms of engineering design objectives, such a system would be designed for minimum CO_2 emissions. However, a clean system can only be feasible if it is implemented at a reasonable cost. Therefore, the primary design considerations for this application are to minimize total CO_2 emissions while keeping the net electricity cost to a minimum. This is a multi-objective optimization problem with two competing design objectives – the solution to which is shown in Figure 5-4 and discussed in §5.2.5.

Using a "trial-and-error" approach on the simulation in Figure 5-1, the principle designer may arrive at a hybrid energy system configuration, which results in the lowest possible CO_2 emissions. This configuration, termed *initial design*, is indicated in Figure 5-8, with all the design variables and objectives summarized in Table 5.4. From the design tradeoffs curve, it is evident that the initial design lies on the bleeding edge of the curve – where a large increase in electricity cost will pay for only a minimum reduction in total CO_2 emissions. To improve this design, the principal designer scans the design objective space and selects a more favourable set of optimal tradeoffs to be passed to the system model as design targets. The system model inputs corresponding to these design targets are used to rapidly derive this design. The process is illustrated in Figure 5-8 and the new design, termed *final design*, is summarized in Table 5.4.

Providing Affordable Electricity to the Island's Inhabitants

Local governments in remote locations may choose to subsidize the purchase of hybrid energy systems to provide affordable electricity for their constituents. Benefits of such a program would include reduced environmental impact and improved public health – if the installed systems would drastically reduce the need for polluting fossil fuels as an energy source. However, to make this practice sustainable, these systems must be produced cost effectively. Therefore, the primary design considerations for this application are to minimize the net electricity cost while keeping the total

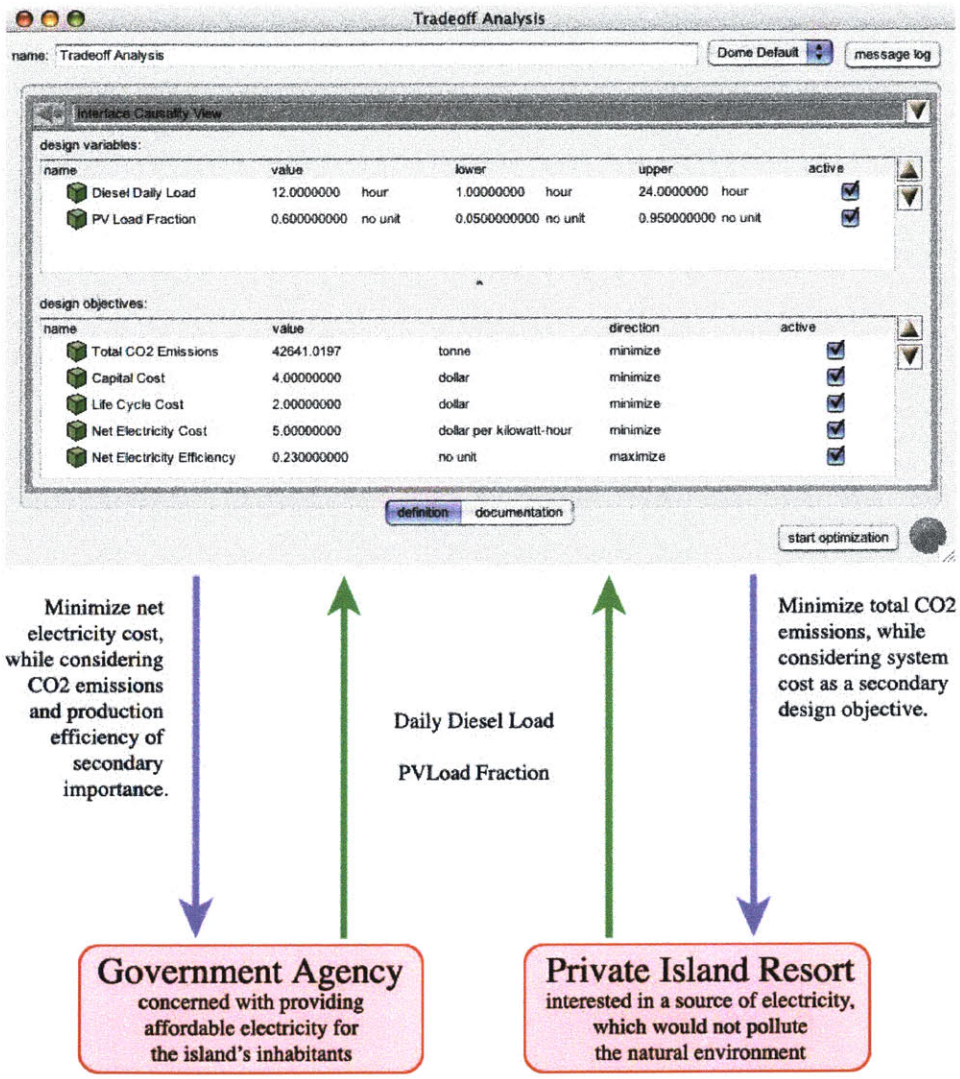
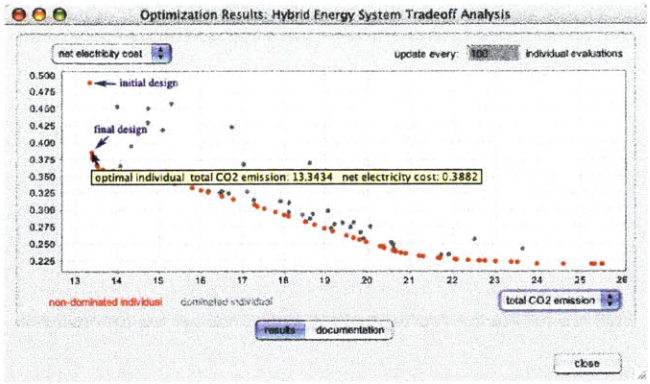
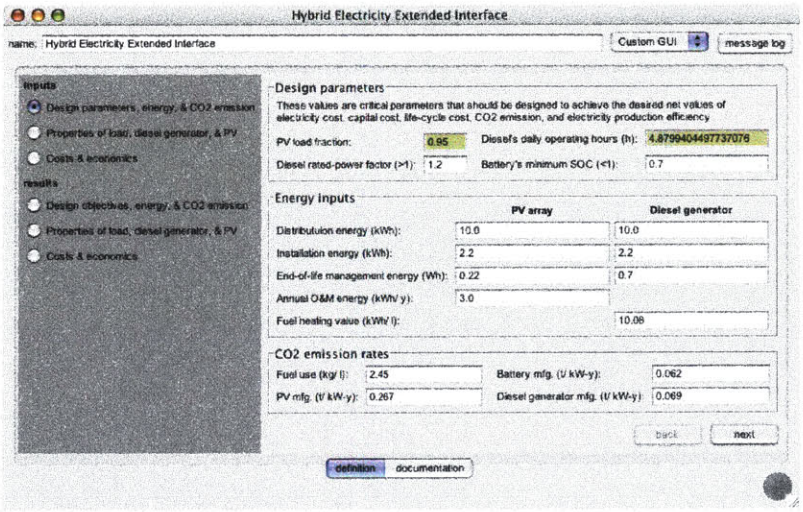


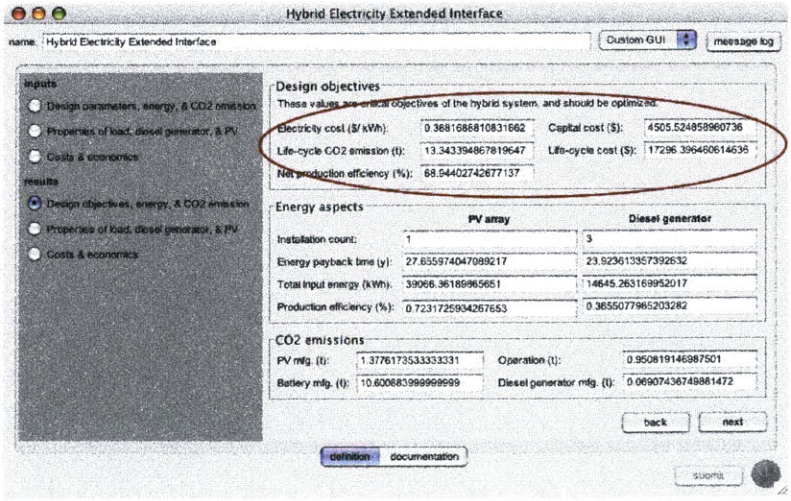
Figure 5-7: Individuals interacting with the hybrid energy system optimization service. Each individual is designing the system for a different application with a unique set of prioritized design objectives.



(a)



(b)



(c)

Figure 5-8: (a) Designer selects optimal tradeoffs as design targets. (b) Inputs are passed to the system model (marked in green). (c) New "optimal" design is derived with the design objectives circled in red.

Parameter	Initial	Final
Variables:		
PV load fraction	0.95	0.95
total daily diesel operation	20.75	4.88
Objectives:		
net electricity cost	0.49	**0.39**
total CO_2 emission	13.29	13.34
capital cost	3948.63	4505.52
life cycle cost	43555.19	**17296.40**
net electricity efficiency	67.96	**68.94**

Table 5.4: Initial and final design of the hybrid energy system – optimized for the private island resort application. Units for each respective parameter are given in Figure 5-3. Improved design objectives are indicated in bold.

CO_2 emissions to a minimum. Furthermore, the energy system should operate at the maximum electricity production efficiency. Solutions, illustrating tradeoffs between the design objectives, were presented in Figure 5-4, Figure 5-5, Figure 5-6 and discussed in §5.2.5.

Similar to the previous example, the principle designer for this application may use the simulation in Figure 5-1 to derive, using a "trial-and-error" approach, a hybrid energy system configuration, which results in the lowest possible net electricity cost. This *initial design* is indicated in Figure 5-9 and summarized in Table 5.5. Although this design produces the least expensive electricity, it is far from being optimal. The design tradeoff curve in Figure 5-4 indicates that significant reductions in CO_2 emissions could be made at only a small increase in electricity cost, and so the principal designer selects a new set of performance targets, marked *intermediate design* in Figure 5-9. This newly proposed design configuration greatly reduces the total CO_2 emissions, however, it still produces electricity with low efficiency. As a result, the tradeoff curve in Figure 5-10 is used to select a new set of design targets, which significantly improve the electricity production efficiency and reduce total CO_2 emissions, while keeping the increase in net electricity cost to a minimum. The final design is summarized in Table 5.5.

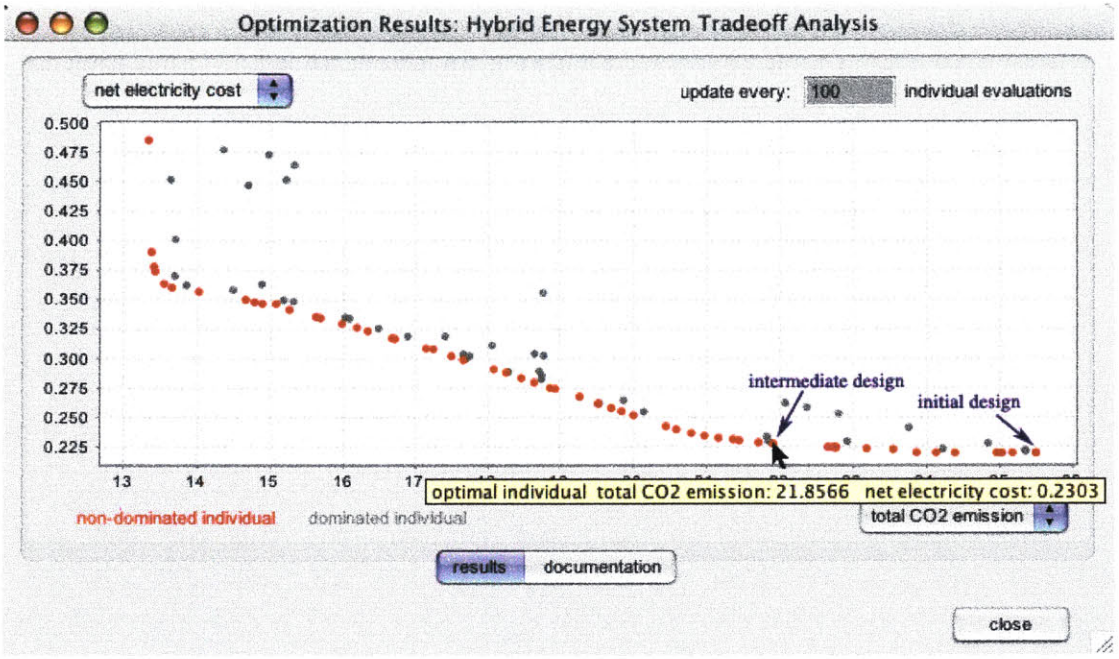


Figure 5-9: Initial and intermediate hybrid energy system designs for the governmental subsidy program application. Designs are compared based on net electricity cost and total CO_2 emissions.

Parameter	**Initial**	**Intermediate**	**Final**
Variables:			
PV load fraction	0.05	0.05	0.52
total daily diesel operation	1	2.27	1.0
Objectives:			
net electricity cost	0.22	0.23	0.30
total CO_2 emission	25.45	21.86	**19.23**
capital cost	6199.31	4968.56	6515.03
life cycle cost	52044.17	54055.05	**31197.03**
net electricity efficiency	38.54	38.54	**50.19**

Table 5.5: Initial and final design of the hybrid energy system – optimized for the government subsidy program application. Units for each respective parameter are given in Figure 5-3. Improved design objectives are indicated in bold.

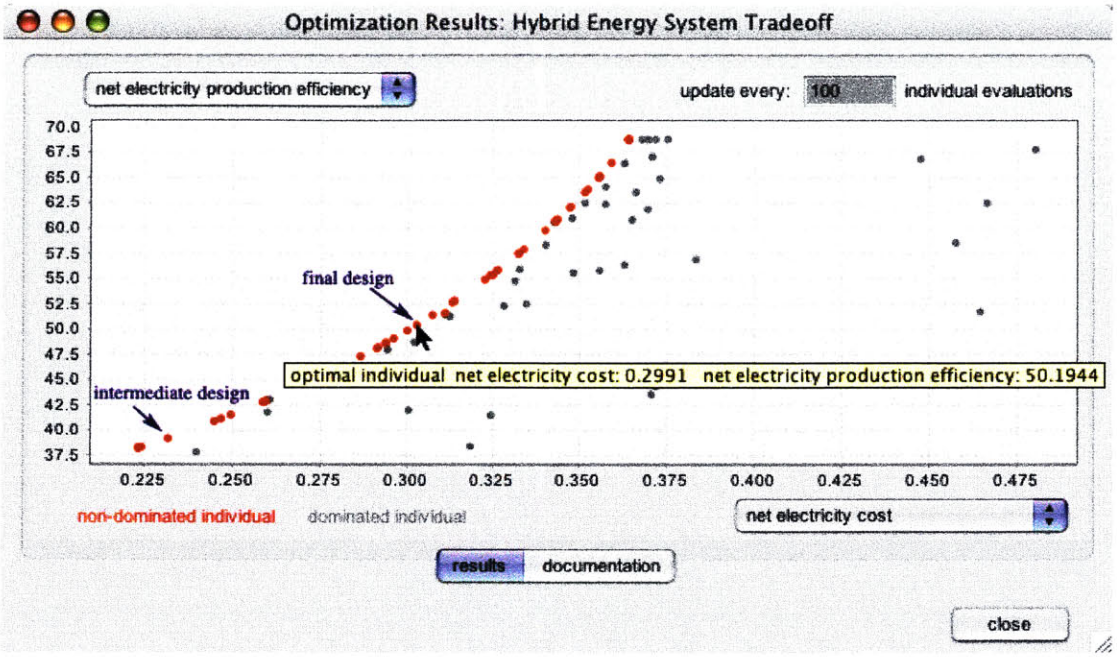


Figure 5-10: Intermediate and final hybrid energy system designs for the governmental subsidy program application. Designs are compared based on net electricity production efficiency and electricity cost.

5.3 Summary

The hybrid PV-diesel energy system design scenario, originally developed by Sukkasi [70], was used in this section to validate the target-driven design approach. Consisting of six independent modules, the integrated simulation predicts the total life-cycle cost, net electricity cost, CO_2 emission, capital cost and net electricity production efficiency for a given set of inputs – namely the daily diesel operation time and PV load fraction. The simulation was optimized, using the tool developed in this work and discussed in §4.3. The algorithm was configured to a population size of 100 individuals and a total number of 1000 evaluations. Each individual took approximately 5 seconds to evaluate. Results of the optimization found optimal design tradeoffs between the following sets of competing design objectives: (i) net electricity cost vs. total CO_2 emission, (ii) net electricity production efficiency vs. net electricity cost, and (iii) net electricity production efficiency vs. total CO_2 emission. The results were used to provide optimal design configurations of a hybrid PV-diesel energy system for two unique applications: (i) a private island resort, and (ii) a governmental subsidy program for providing affordable electricity. It was shown that using the design tool developed in

this work, individuals running simulations can quickly select optimal tradeoffs between competing design objectives as design targets to the system model to derive a new, and optimal, system model state. For example, for the private resort application, a set of design targets was chosen that resulted in a net electricity cost reduction of 20% with only a 1% increase in total CO_2 emission. Similar results were achieved for the governmental subsidy program application.

Chapter 6

Conclusion

6.1 Summary

Chapter 2 provides the background for the design tool developed in this work. The discussion is broken down to four parts. First, an overview of the traditional product development process is presented. Organizations participating in such an activity face two common challenges: (i) creating an integrated product model, and (ii) resolving design tradeoffs during product development. To address the first challenge, a number of integrated product modeling tools have been developed commercially and in academia. DOME, an integrated product modeling framework used in this work, was discussed in detail. The tool developed in this work utilizes evolutionary algorithms, which were discussed in this chapter along with several other optimization algorithms. Finally, an overview of formal multi-disciplinary optimization methods is provided and several commercially available packages offering such capabilities are presented.

Chapter 3 provides a detailed discussion of the optimization algorithm, QMOO, which was implemented in the design tool developed in this work. QMOO is an evolutionary algorithm originally developed for optimizing energy system problems. It has a non-generational population structure, where individuals are added and removed from the population as needed. The algorithm is elitist – where its population contains only the best individuals found so far. This approach ensures rapid convergence, however elitist algorithms may create populations which lack diversity, resulting in converge to some local optima before fully exploring the problem's design space. To address this issue, QMOO divides the population into groups in their parameter space and allows each group to evolve independently. QMOO implements Pareto-based multi-objective optimization and has an

asynchronous parallel structure. Asynchronous algorithm structures use computational resources more effectively and are discussed in chapter 3. QMOO was chosen for its ability to handle discontinuous design spaces and multi-objective problems, which are encountered in most practical engineering problems. Furthermore, the algorithm's architecture lends itself to evaluation of objective functions in parallel. Although the parallelism feature of the algorithm has not been exploited by this author, future implementations of this design tool should do so.

Chapter 4 describes the developed design tool and discusses its implications on the traditional product development process. A trivial "tube-bundle heat exchanger" design scenario is introduced to demonstrate the use of this design tool in its four modes: (i) building a model, (ii) publishing a model on a web-based server, (iii) executing the underlying simulation and, (iv) analyzing its results. Also discussed, is how the design tool evaluates and resolves design tradeoffs in an integrated product model. A methodology has been introduced, which allows individuals subscribing to an interface of an integrated product model to use optimal tradeoffs, between two design objectives, as design targets to configure that model so as to achieve optimal overall product performance. The next section of this chapter focuses on the communication architecture between the optimization tool and its DOME framework. Two Java objects have been developed for this purpose: (i) evaluator, and (ii) monitor. The evaluator object is responsible for passing variable and objective function information between the optimization engine and the integrated product model. The monitor object, as the name implies, is responsible for monitoring the simulation and sending information about the solution to the interface client. The final section of this chapter discusses, in detail, the implications of this work on traditional approaches to engineering design. In summary, it allows individuals to run many more design improvement cycles, rapidly evaluating and elucidating design tradeoffs at each iteration. Through the "target-driven" design approach, individuals can also derive product model configurations with optimal overall product performance.

In Chapter 5, application of this work to the design of a hybrid PV-diesel energy system is presented. The integrated system model consisted of six independent modules predicting the total life-cycle cost, net electricity cost, CO_2 emission, capital cost and net electricity production efficiency. Using the design tool developed in this work, design tradeoffs were studied for the following sets of competing design objectives: (i) net electricity cost vs. total CO_2 emission, (ii) net electricity production efficiency vs. net electricity cost, and (iii) net electricity production efficiency vs. total CO_2

emission. The results of these studies were used to provide energy systems optimally configured for two independent applications – a private island resort and a governmental electricity subsidy program. This application demonstrates that the developed design tool is effective at resolving design tradeoffs and deriving system model configurations that maximize the overall product performance.

6.2 Future Work

A number of additional features should be implemented in the design tool to further improve its effectiveness and usability.

The first set of tasks would be to improve the overall functionality and robustness of the optimization engine used by the analysis tool. Currently, only one multi-objective evolutionary algorithm has been implemented. Although this is sufficient for a large number of engineering problems, there are cases where such an algorithm is unsuitable. For example, design problems with a single objective function. In the future, the tool should support multiple optimization techniques and the choice of which to use for a particular problem should be left to the user. Furthermore, the current configuration panel could be enhanced to allow users with an advanced knowledge of optimization techniques more control in configuring an algorithm for a particular optimization problem. Finally, at the moment the design tool only supports real variables. In the future, more data types should be implemented in the design tool to allow for application to a wider range of engineering problems.

In addition, an API for the design tool should be developed to allow interaction with optimization models through a third party application or a simple Java subroutine. It is envisioned, for example, that such an API would allow models to be published on the Internet using an appropriate server technology. Users would then be able to interact with the model interface in a web browser without having to use the DOME client application. At the time of completion of this paper, work in this area has been initiated and considerable progress has been made.

The design tool discussed in this work, is the first of its type developed for the DOME framework. If more analysis tools are to be developed in the future, a general architecture, or recipe, for their implementation should be in place. The architecture should provide general object templates, which could be customized or reused for individual tools. Finally, further work should be done in the area

of testing and improving the general robustness of this tool as the scale and complexity of integrated models increases. It is the experience of this author, that integrations involving models from multiple third-party applications would often crash during a simulation. If the objective functions take minutes to evaluate, such events can be taxing on computational resources. Therefore, it would be prudent to have a mechanism, which would allow the user to restart the simulation at exactly the point where it crashed. Currently, such mechanisms do not exist.

6.3 Final Words

Product development is a difficult task and few organizations are successful at it. Some of the challenges include a thorough exploration of all design alternatives, as well as the elucidation and resolution of design tradeoffs. Developing a design tool that would address these challenges was the mission of this work.

It is the ultimate hope of this author that this work will help product development teams develop better products.

Appendix A

Pseudo-Code for the Evaluator and Monitor Communication Objects

The Evaluator and Monitor objects are classes that enable communication between DOME and the optimization engine QMOO. The Evaluator object is used by the optimization engine to "interrogate" the integration model, while the Monitor objet is used to pass information about the solution to the client.

The pseudo code for each object will now be discussed.

A.1 The Evaluator Object

```
public DomeEvaluator(OptimizationRuntime object)
{
    assign the optimization runtime object to a member variable of this class
    call obtainVariablesAndObjectives()
}

void obtainVariablesAndObjectives()
{
    get a list of all variables in the optimization runtime object
    while(list not empty)
    {
        if(variable in the optimization runtime object is also in the interface object && variable is active)
        {
            add variable to the list of variables in the evaluator object
        }
    }
    get a list of all objectives in the optimization runtime object
    while(list not empty)
    {
        if(objective in the optimization runtime object is also in the interface object && objective is active)
        {
            add objective to the list of objectives in the evaluator object
        }
    }
}

void start(Objectives Map, Variables Map)
{
    /* This method is called by the optimization algorithm QMOO, once at the beginning of each
    optimization run. By the time this method is called, the objective and variable lists in the evaluator object
```

```
    would have been populated. This method is used by the optimization engine to obtain information about
    the number of objectives and the decision variable space.*/
    in the objectives map, set the number of design objectives
    in the variables map, set the number of design variables
    for (each design variable)
    {
        set the data object type
        set the upper and lower search limit
    }
}

void process(EvaluatorData object)
{
    /* This method is called by the optimization algorithm whenever a new individual needs to be evaluated.
    It passes a QMOO native EvaluatorData object, which contains information about the individual under
    evaluation.*/
    if(current project thread is null)
    {
        start new project thread and pass into it, the current individual contained in the EvaluatorData object
    }
    else
    {
        if(project thread is done)
        {
            indicate to the EvaluatorData object that the current individual has been evaluated
            set the project thread to null
        }
    }
}

class ProjectThread
{
```

```
    /*This is the thread in which the integration project is solved to obtain corresponding objective function
    values for an individual with a particular design vector configuration.*/
    public ProjectThread(Individual)
    {
        assign individual passed in, to the current individual in the project thread object
        create project status listener
    }

    void run()
    {
        for(each variable in the evaluator variables list)
        {
            set the input variables to the integration project with the design vector of the current individual
        }
        run project
    }

    ProjectStatusListner()
    {
        if(integration project is solved)
        {
            set objective values in current individual
        }
    }
}
```

A.2 The Monitor Object

```
public Monitor(OptimizationRuntime object)
{
    set optimization runtime object member variable to the current object passed in
}

void go(Monitor object)
{
    /*This method is called by the optimization engine at some user-specified number of evaluations interval.*/
    from the Monitor object, get current population
    while (population size is not 0)
    {
        send individual to interface client
    }
}
```

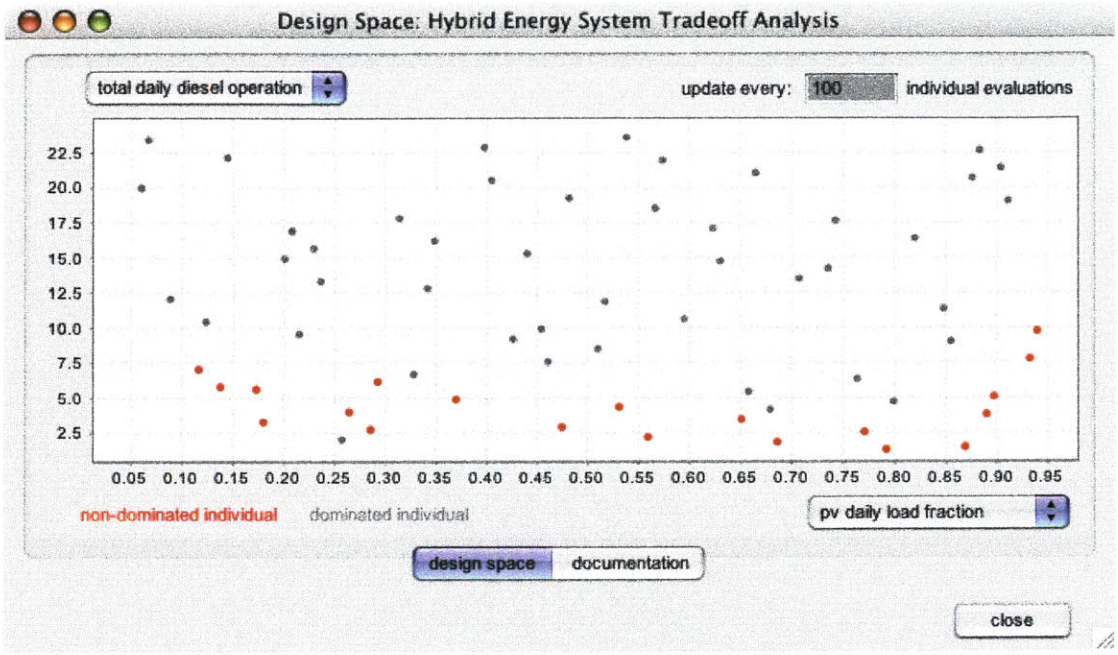
Appendix B

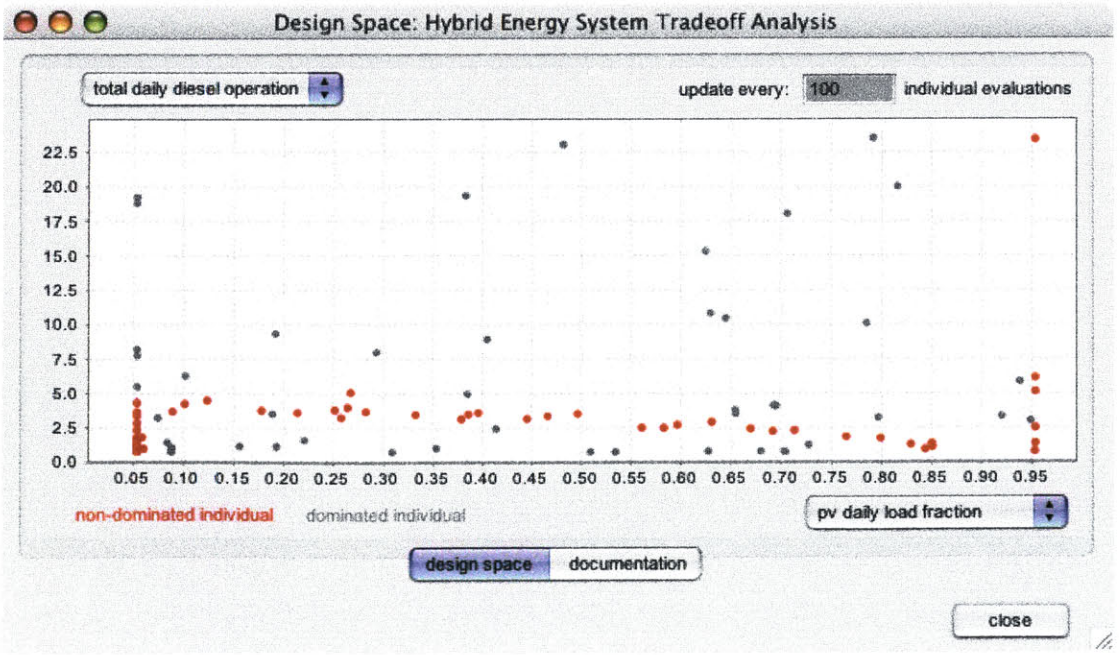
Hybrid PV-Diesel Energy System Decision Variable Plots

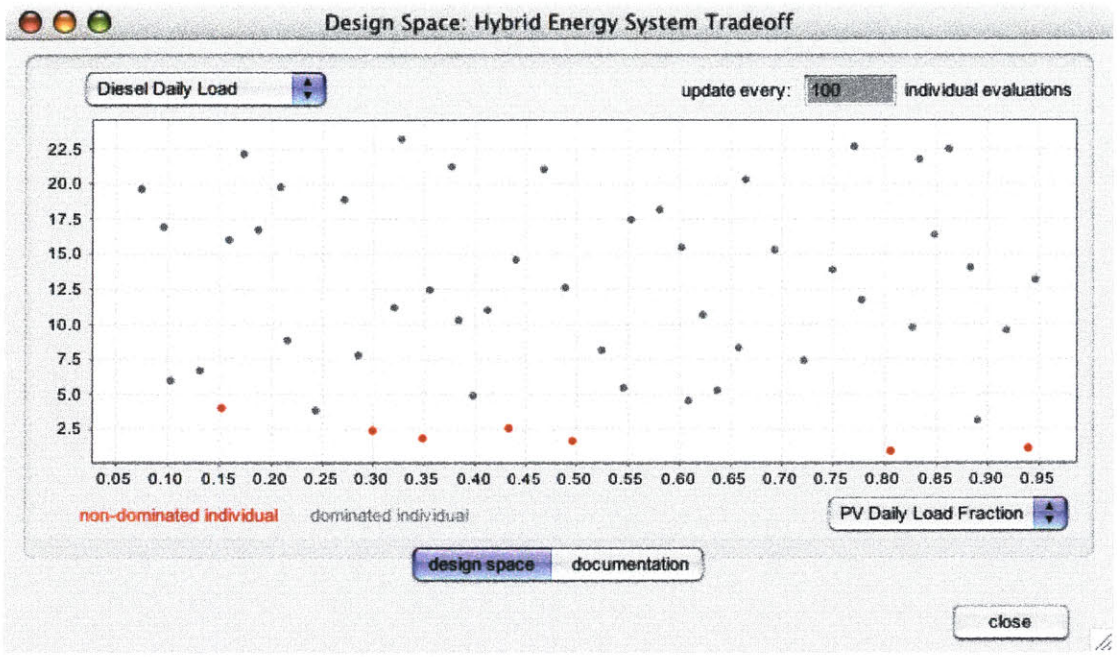
Population of the hybrid pv-diesel energy system optimization problems graphed in the decision variable space. Graphs appear in the following order of executed simulations:

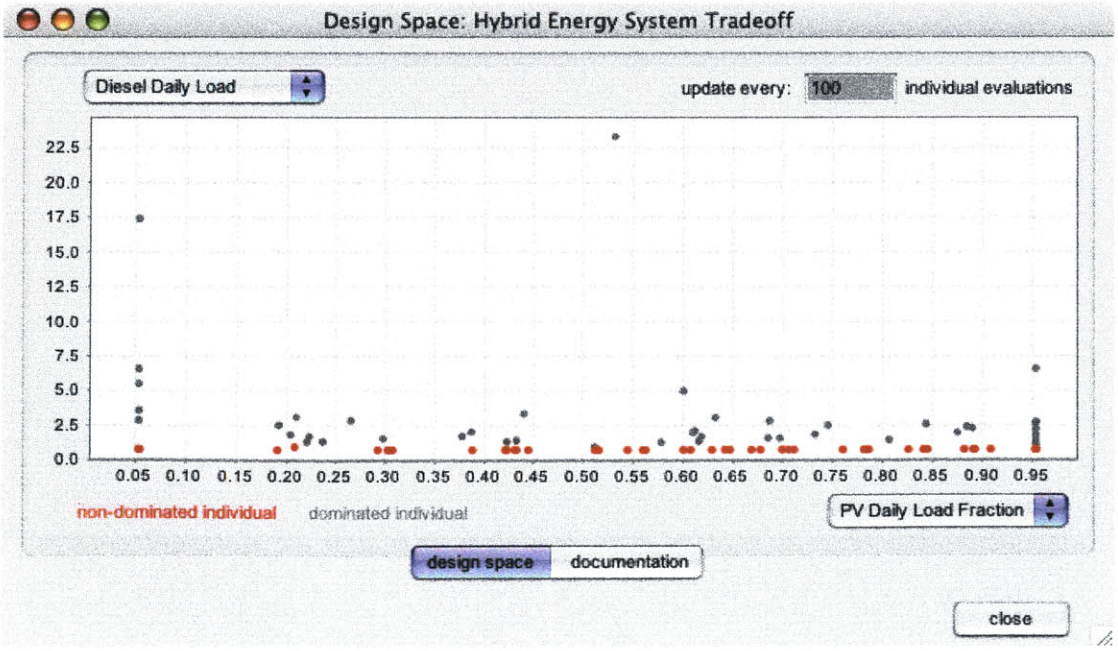
- net electricity cost vs. total CO_2 emission
- net electricity production efficiency vs. net electricity cost
- net electricity production efficiency vs. total CO_2 emission

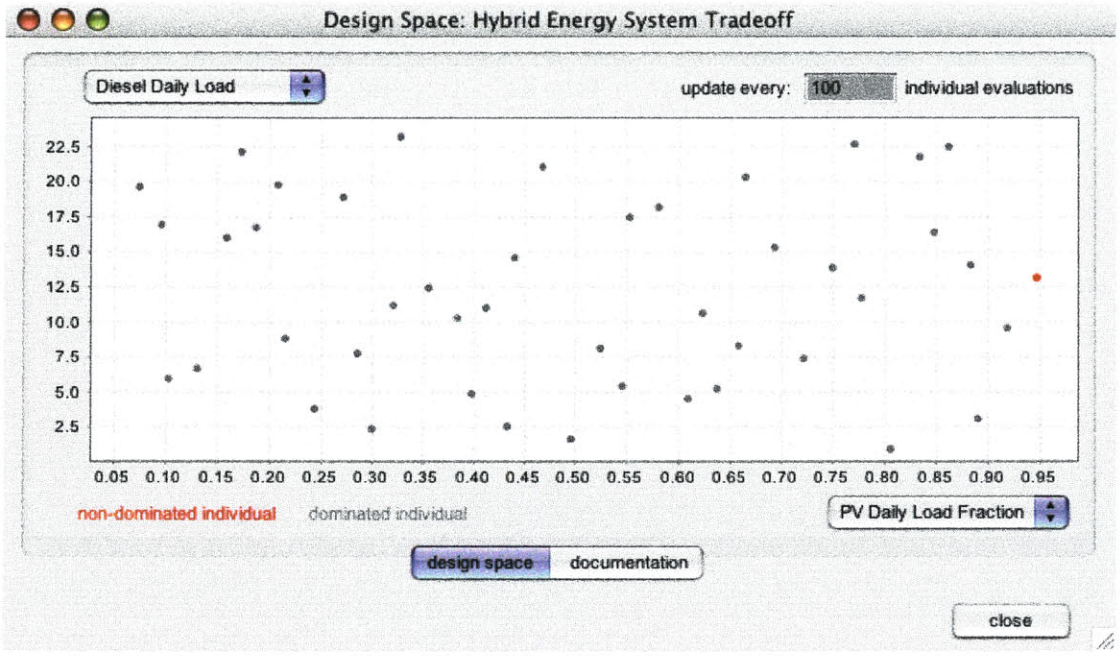
In each set of results, the top graph indicates the population after 100 evaluations, while the bottom graph illustrates the populations after 1000 evaluations.

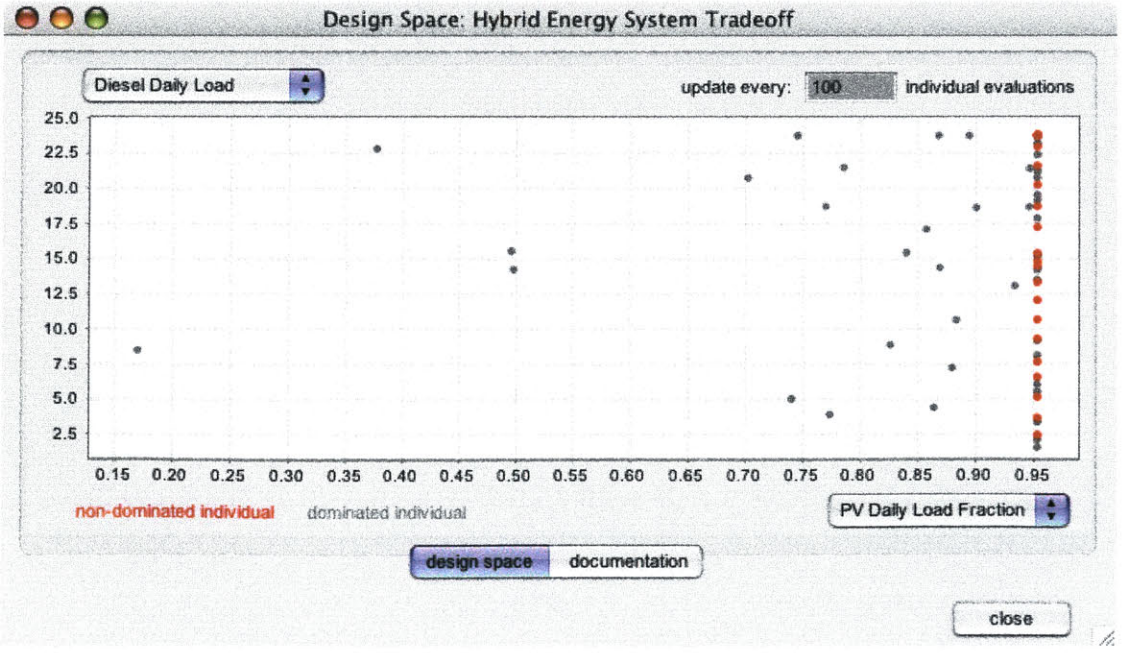

Design Space: Hybrid Energy System Tradeoff Analysis
total daily diesel operation
update every: 100 individual evaluations
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50 0.55 0.60 0.65 0.70 0.75 0.80 0.85 0.90 0.95
non-dominated individual
dominated individual
pv daily load fraction
design space
documentation
close


Design Space: Hybrid Energy System Tradeoff Analysis
total daily diesel operation
update every: 100 individual evaluations
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50 0.55 0.60 0.65 0.70 0.75 0.80 0.85 0.90 0.95
non-dominated individual
dominated individual
pv daily load fraction
design space
documentation
close


Design Space: Hybrid Energy System Tradeoff
Diesel Daily Load
update every: 100 individual evaluations
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50 0.55 0.60 0.65 0.70 0.75 0.80 0.85 0.90 0.95
non-dominated individual
dominated individual
PV Daily Load Fraction
design space
documentation
close


Design Space: Hybrid Energy System Tradeoff
Diesel Daily Load
update every: 100 individual evaluations
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50 0.55 0.60 0.65 0.70 0.75 0.80 0.85 0.90 0.95
non-dominated individual
dominated individual
PV Daily Load Fraction
design space
documentation
close


Design Space: Hybrid Energy System Tradeoff
Diesel Daily Load
update every: 100 individual evaluations
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50 0.55 0.60 0.65 0.70 0.75 0.80 0.85 0.90 0.95
non-dominated individual
dominated individual
PV Daily Load Fraction
design space
documentation
close


Design Space: Hybrid Energy System Tradeoff
Diesel Daily Load
update every: 100 individual evaluations
25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50 0.55 0.60 0.65 0.70 0.75 0.80 0.85 0.90 0.95
non-dominated individual
dominated individual
PV Daily Load Fraction
design space
documentation
close

References

[1] J. Alander and J. Lampinen. Cam Shape Optimisation by Genetic Algorithm. In D. Quagliarella, J. Périaux, C. Polini, and G. Winter, editors, *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science*, pages 153–174. Wiley, Chichester, 1998.

[2] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.

[3] R. J. Balling and J. Sobieszczanski-Sobieski. Optimization of Coupled Systems: A Critical Overview of Approaches. *AIAA Journal*, 34(1):617, 1996.

[4] R. Banares-Alcantara. Representing the Engineering Design Process: Two Hypotheses. *Computer-Aided Design*, 23(9):595–603, 1991.

[5] P. I. Bliznakov, J. J. Shah, D. K. Jeon, and S. D. Urban. Design Information System Infrastructure to Support Collaborative Design in a Large Organization. In *Proceedings of ASME DETC*, vol. 1, pages 1–8, Boston, MA, 1995.

[6] R. D. Braun and I. Kroo. Development and Application of the Collaborative Optimization Architecture in a Multidisciplinary Design Environment. In N. Alexandrov and M. Y. Hussaini, editors, *Multidisciplinary Design Optimization, State of the Art*. SIAM, 1997.

[7] C. G. Broyden. Quasi-Newton Methods and their Application to Function Minimization. *Mathematics of Optimization*, 21:368–381, 1967.

[8] A. Cabraal, A. M. Cosgrove-Davies, and L. Shaeffer. Accelerating Sustainable Photovoltaic Market Development. *Progr. Photovolt. Res. Appl.*, 6:297–306, 1998.

[9] J. Cagan and C. M. Vogel. *Creating Breakthrough Products*. Prentice Hall, Englewood Cliffs, New Jersey, 2002.

[10] Q. Cao and D. R. Walace. Distributed Solving to Support Emergent Behaviour of DOME. Technical report, MIT CADlab, 2004.

[11] R. K. Carrol and G. E. Johnson. Approximate Equations for the AGMA J-factor. *Mechanisms and Machine Theory*, 23(6):449–450, 1960.

[12] M. P. Case and S. C. Y. Lu. Discourse Model for Collaborative Design. *Computer-Aided Design*, 28(5):335–345, 1996.

[13] A. Cauchy. Methode génerale pour la résolution des systéms d'equations simultanées. *Acad. Sci. Paris*, 25:536–538, 1847.

[14] L. D. Chambers. *Practical Handbook of Genetic Algorithms: Complex Coding Systems Volume III.* CRC Press, Boca Raton, FL, 1999.

[15] E. K. P. Chong and S. H. Zak. *An Introduction to Optimization.* John Wiley & Sons, New York, second edition, 2001.

[16] C. A. Coello Coello. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems*, 1(3):269–308, 1999. URL `citeseer.nj.nec.com/coello98comprehensive.html`.

[17] C. A. Coello Coello. An Updated Survey of Evolutionary Multiobjective Optimization Techniques: State of the Art and Future Trends. *Knowledge and Information Systems*, 1(3), 1999.

[18] S. Colle, S. L. Abreu, and R. Ruther. Economic evaluation and optimization of hybrid diesel/photovoltaic systems integrated to utility grids. *Solar Energy*, 76:295–299, 2004.

[19] E. J. Cramer, J. E. Dennis, P. D. Frank, R. M. Lewis, and G. R. Shubin. Problem Formulation for Multidisciplinary Design Optimization. *SIAM Journal on Optimization*, 4(4):754–776, 1994.

[20] M. R. Cutkosky, G. Olsen, J. Tenenbaum, and T. Gruber. Collaborative Engineering Based on Knowledge Sharing Agreements. In *Proceedings of the 1994 ASME Database Symposium*, 1994. URL `http://www.citeseer.ist.psu.edu/olsen94collaborative.html`.

[21] W. C. Davidon. Variance Algorithms for Minimization. *Computer Journal*, 10:406–410, 1968.

[22] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms.* Wiley, Chichester, 2001.

[23] K. Deb and R. Agrawal. Simulated Binary Crossover for Continuous Search Space. *Complex Systems*, 9(2):115–148, 1995. URL `citeseer.nj.nec.com/deb95simulated.html`.

[24] R. J. Eggert. *Engineering Design.* Prentice Hall, Englewood Cliffs, New Jersey, 2005.

[25] L. J. Eschelman and J. D. Schaffer. Real-coded genetic algorithms and interval schemata. In *Foundations of Genetic Algorithms 2*, pages 187–202. Morgan Kaufmann, San Francisco, CA, 1993.

[26] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques.* John Wiley & Sons, New York, 1968.

[27] R. Fletcher. A New Approach to Variable Metric Algorithms. *Computer Journal*, 13:317–322, 1970.

[28] R. Fletcher and M. J. D. Powell. A Rapidly Convergent Descent Method for Minimization. *Computer Journal*, 6:163–168, 1963.

[29] L. J. Fogel. Autonomous Automata. *Industrial Research*, 4:14–19, 1962.

[30] D. D. Frey, F. Engelhardt, and E. M. Greitzer. A Role for One-Factor-at-a-Time Experimentation in Parameter Design. *Research in Engineering Design*, 14:65–74, 2003.

[31] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[32] D. Goldfarb. A Family of Variable Metric Methods Derived by Variational Means. *Math. Comput.*, 24:23–26, 1970.

[33] B. S. Gottfried and J. Weisman. *Introduction to Optimization Theory.* Prentice Hall, Englewood Cliffs, New Jersey, 1973.

[34] M. Hardwick and D. Spooner. An Information Infrastructure for a Virtual Manufacturing Enterprise. In *Proceedings of Concurrent Engineering: A Global Perspective*, pages 417–429, McLean, VA, 1995.

[35] M. R. Hestenes. *Conjugate Direction Methods in Optimization.* Springer-Verlag, Heidelberg, 1980.

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

[37] J. Horn, N. Nafpliotis, and D. E. Goldberg. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, Piscataway, New Jersey, 1994. IEEE Service Center. URL `citeseer.nj.nec.com/horn94niched.html`.

[38] B. Hyman. *Fundamentals of Engineering Design.* Prentice Hall, Englewood Cliffs, New Jersey, 1998.

[39] C. D. Jilla and O. de Weck. Lecture: Simulated Annealing. *16.888: Multidisciplinary System Design Optimization (MSDO)*, March 5th, 2004.

[40] J. E. Kelley. The Cutting Plane Method for Solving Convex Programs. *J. SIAM*, pages 703–713, 1960.

[41] J. B. Kim and D. R. Wallace. A Goal-Oriented Design Evaluation Model. In *Proceedings of the 1997 ASME Design Engineering Technical Conference*, pages 1–9, Sacramento, California, 1997. ASME.

[42] S. Kodiyalam and J. Sobieszczanski-Sobieski. Multidisciplinary design optimization - some formal methods, framework requirements, and application to vehicle design. *Int. J. Vehicle Design (Special Isssue)*, pages 3–22, 2001.

[43] S. Kodiyalam, R. J. Yang, L. Gu, and C. H. Tho. Large-scale, multidisciplinary optimization of a vehicle system in a scalable, high performance computing environment. Technical report, FORD Research Laboratory, Safety Research and Development Department, 1999.

[44] S. Krikpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4593):671–680, 1983.

[45] J. W. Lewis and K. J. Singh. Electronic Design Notebooks (EDN): Technical issues. In *Proceedings of Concurrent Engineering: A Global Perspective*, pages 431–436, McLean, VA, 1995.

[46] G. Leyland. *Multi-Objective Optimization Applied to Industrial Energy Problems*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne, 2002.

[47] T. Markvart. *Solar Electricity*. John Wiley & Sons, England, second edition, 2000.

[48] A. F. Mills. *Heat Transfer*. Prentice Hall, Upper Saddle River, New Jersey, Second edition, 1999.

[49] D. C. Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons, New York, 2001.

[50] R. H. Myers and D. C. Montgomery. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. John Wiley & Sons, New York, 1995.

[51] S. I. Newton. *Universal Arithmetick: or, A Treatise of Arithmetical Composition and Resolution*. J. Senex et al., London, 1720.

[52] J. Owen. *STEP - An Introduction*. Winchester, 1993.

[53] K. F. Pahng, N. Senin, and D. R. Wallace. Distributed Modeling and Evaluation of Product Design Problems. *Computer-Aided Design*, 30(6):411–423, 1998.

[54] K. F. Pahng, N. Senin, and D. R. Wallace. Modeling and Evaluation of Product Design Problems in a Distributed Design Environment. In *Proceedings of ASME DETC'97*, Sacramento, California, 1998.

[55] P. Y. Papalambros and D. J. Wilde. *Principles of Optimal Design: Modeling and Computation*. Cambridge University Press, Cambridge, UK, Second edition, 2000.

[56] F. Pena-Mora, D. Sriram, and R. Logcher. SHARED DRIMS: SHARED Design Recommendation-Intent Management System. *Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 213–221, 1993.

[57] F. Pena-Mora, D. Sriram, and R. Logcher. Conflict Mitigation System for Collaborative Engineering. *EDAM - Special Issue of Concurrent Engineering*, 9(2):101–123, 1995.

[58] C. Petrie, M. R. Cutkosky, and H. Park. Design Space Navigation as a Collaborative Aid. In *Proceedings of Third International Conference on Artificial Intelligence in Design*, Lausanne, Switzerland, 1993.

[59] M. S. Phadke. *Quality Engineering Using Robust Design*. Prentice Hall, Englewood Cliffs, New Jersey, 1989.

[60] W. K. Purves, D. Sadava, G. H. Orians, and H. C. Heller. *Life: The Science of Biology*. Sinauer Associates, Inc., Sunderland, MA, sixth edition, 2001.

[61] G. Schramm. Accelerating Photovoltaic Production Through Grid Connected Applications in Developing Countries. In *Proceedings of the 28th IEEE Photovoltaic Specialists Conference*, Anchorage, USA, 2000.

[62] H.-P. Schwefel. Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik. Diplomarbeit, Technische Universität Berlin, 1965.

[63] N. Senin, D. R. Wallace, and N. Borland. Distributed Object-based Modeling in Design Simulation Marketplace. *ASME Journal of Mechanical Design*, 125:2–13, 2003.

[64] D. F. Shanno. Conditioning of Quasi-Newton Methods for Function Minimization. *Math. Comput.*, 24:647–656, 1970.

[65] J. Sobieszczanski-Sobieski. Optimization by Decomposition: A Step from Hierarchic to Non-hierarchic Systems. In N. CP-3031, editor, *Proceedings, 2nd NASA/USAF Symposium on Recent Advances in Multidisciplinary Analysis and Optimization*, Hampton, Virginia, 1988.

[66] J. Sobieszczanski-Sobieski, J. Agte, and J. R. Sandusky. Bi-Level Integrated System Synthesis (BLISS). In *Proceedings, 7th AIAA/USAF/NASA/iSSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, Missouri, Steptember, 1998. AIAA.

[67] J. Sobieszczanski-Sobieski and R. T. Haftka. Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments. *Structural Optimization*, 14(1):123, 1997.

[68] J. Sobieszcznski-Sobieski. Optimization by Decomposition: A Step from Hierarchic to Non-hierarchic Systems. In *Proceedings, 2nd NASA/USAF Symposium on Recent Advances in Multidisciplinary Analysis and Optimization*, volume NASA CP-3031, Hampton, Virginia, 1988.

[69] D. Sriram and R. Logcher. The MIT DICE project. *IEEE Computer*, pages 64–65, 1993.

[70] S. Sukkasi. Alternative Energy Design Toolkit. Master's thesis, Massachusetts Institute of Technology, 2004.

[71] G. Toye, M. R. Cutkosky, J. Tenenbaum, and J. Glicksman. SHARE: A Methodology and Environment for Collaborative Product Development. In *Proceedings of Second Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 33–47, Morgantown, West Virginia, 1993.

[72] K. T. Ulrich and S. D. Eppinger. *Product Design and Development*. McGraw-Hill, New-York, Third edition, 2004.

[73] D. A. Van Veldhuizen and G. B. Lamont. Multiobjective Evolutionary Algorithm Test Suites. In J. Carroll, H. Haddad, D. Oppenheim, B. Bryant, and G. B. Lamont, editors, *Proceedings of the 1999 ACM Symposium on Applied Computing*, pages 351–357, San Antonio, Texas, 1999. ACM. URL `citeseer.nj.nec.com/david99multiobjective.html`.

[74] G. N. Vanderplaats. *Numerical Optimization Techniques for Engineering Design with Applications*. McGraw-Hill, New-York, 1984.

[75] D. E. Veley. Optimization in the Adaptive Modeling Language. AIAA 98-4872, Air Force Research Laboratory, Structures Division, 1998.

[76] D. Wallace, E. Yang, and N. Senin. Integrated simulation and design synthesis. Technical report, MIT CADlab, 2003.

[77] D. R. Wallace, S. Abrahamson, N. Senin, and P. Sferro. Integrated Design in a Service Marketplace. *Computer-Aided Design*, 32(2):97–107, 2000.

[78] A. W. Westerberg, R. Coyne, D. Cuningham, A. Dutoit, E. Gardner, S. Konda, S. Levy, I. Monarch, R. Patrick, Y. Reich, E. Subrahmanian, M. Terk, and M. Thomas. Distributed and Collaborative Computer-Aided Environment in Process Engineering Design. In *Proceedings of ISPE*, 1995.

[79] R. Xiaojuan, P. Zhelong, R. Eigenmann, and Y. C. Hu. Decentralized and Hierarchical Discovery of Software Applications in the iShare Internet Sharing System. In *Proceedings of International Conference on Parallel and Distributed Computing Systems*, San Francisco, California, 2004.

[80] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimisation: Methods and Applications*. PhD thesis, Eidgenöissiche Technische Hochschule Zürich, Nov. 1999.