NUWC-NPT Technical Report 11,741
15 May 2006

The Genetic-Algorithm-Based Normal Boundary Intersection (GANBI) Method: An Efficient Approach to Pareto Multiobjective Optimization for Engineering Design

Thomas A. Wettergren
Combat Systems Department

NEWPORT

**Naval Undersea Warfare Center Division
Newport, Rhode Island**

Approved for public release; distribution is unlimited.

PREFACE

This work was funded under NUWC Division Newport's In-House Laboratory Independent Research (ILIR) Program.

The technical reviewer for this report was Sandie P. Grage (Code 2532).

Reviewed and Approved: 15 May 2006

[illegible]

Ernest Correia
Head (acting), Combat Systems Department

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE 15 May 2006	3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE The Genetic-Algorithm-Based Normal Boundary Intersection (GANBI) Method: An Efficient Approach to Pareto Multiobjective Optimization for Engineering Design	5. FUNDING NUMBERS
6. AUTHOR(S) Thomas A. Wettergren	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Undersea Warfare Center Division 1176 Howell Street Newport, RI 02841-1708	8. PERFORMING ORGANIZATION REPORT NUMBER TR 11,741
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

A new method for developing tradeoffs in the engineering of complex systems is described. The Genetic-Algorithm-Based Normal Boundary Intersection (GANBI) method serves as a preprocessor for conventional genetic-algorithm-based Pareto optimization solvers. The algorithm is based on applying the normal boundary intersection approach of Pareto optimization to genetic solvers. The approach is shown to provide rapid convergence and to provide a better estimate of the Pareto set than existing state-of-the-art methods. A description of Pareto optimization methods for engineering design is included to put the new method in the context of existing solution approaches. The algorithm for the GANBI method is derived and detailed in the report, and numerical examples showing its efficiency in solving an academic problem are presented. The report concludes with an example of how the GANBI method has been used to make tradeoff decisions in the design of large-scale distributed undersea sensor networks.

14. SUBJECT TERMS Engineering Design Tradeoffs; Genetic Algorithms; Optimal Design GANBI; Pareto Multiobjective Optimization	15. NUMBER OF PAGES 50
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Cont'd)

LIST OF TABLES

THE GENETIC-ALGORITHM-BASED NORMAL BOUNDARY INTERSECTION (GANBI) METHOD: AN EFFICIENT APPROACH TO PARETO MULTIOBJECTIVE OPTIMIZATION FOR ENGINEERING DESIGN

INTRODUCTION

Many complex engineered systems are based on design decisions that involve the resolution of multiple conflicting objectives. The most basic of these is the age-old tradeoff between cost and performance. In such cases, it is desirable to lower costs (a mathematical objective of minimizing the cost function) while also improving performance (a mathematical objective of maximizing the performance function). When the multiple objectives can be simultaneously satisfied in a design, there is no complexity to the decision process. However, practical constraints often create conflict between the simultaneous optimization of these objectives, and this conflict creates a necessary design tradeoff. The study of the development of this tradeoff decision space is referred to as "Pareto multiobjective optimization." This term is used to explicitly note that the optimization goal is to develop an efficient tradeoff set (i.e., set of Pareto solutions) for the optimization process, rather than the delivery of a specific tradeoff decision.

The process of optimal design has been formalized as a discipline of its own in recent years (see Papalambros and Wilde[1] for details). The careful restriction of engineering design problems to the essential parameters for optimization is no longer a heuristic process; it has developed into a formal engineering method. When multiple conflicting objectives occur in a problem, Pareto multiobjective optimization techniques are used to find the efficient tradeoff set, and then knowledgeable decision-makers use this information to make informed design choices. The automation of Pareto decisions has been limited to simplistic problems in the process control industries, where the time scales are relatively slow.[2] Other practical applications must be limited to the presentation of information to the informed decision-makers. A survey of display techniques for such information can be found in Miettinen.[3] In the design of complex systems for undersea warfare, Pareto multiobjective optimization techniques offer the potential for identifying efficient tradeoffs, particularly in new technology areas where there is no vast experience base to permit such judgments without computational aids. Of particular interest to undersea warfare are the "systems of systems," where existing subsystems are interfaced to make larger systems. In these cases, the subsystems are often well-understood, but how their interaction affects the military benefit is often only understood after exercising complex models and/or simulations. Since military benefit is often stated in terms of a performance-risk or performance-cost tradeoff, the necessary optimization is inherently multiobjective, and the shape of the set of Pareto solutions in the multiobjective tradeoff space is of utmost interest in setting design criteria for the subsystems.

This report presents the development of a new method for Pareto multiobjective optimization and shows how it can be used to make difficult design decisions for complex engineered systems. First, the general framework of Pareto multiobjective optimization is described in mathematical terms. From this framework, the pros and cons of the two main classes of Pareto multiobjective optimization methods are described: gradient techniques and evolutionary techniques. This assessment explains the rationale for using Pareto genetic algorithms for providing design tradeoffs in complex engineered systems. The new algorithm—the Genetic-Algorithm-Based Normal Boundary Intersection (GANBI) method—is then described. The GANBI method functions as a preprocessor for existing Pareto genetic solvers. The details of the derivation of the GANBI method are shown, and an illustration is given of its performance relative to standard Pareto genetic solvers on an academic problem. The report concludes by showing an example of the use of the GANBI method in an important problem in undersea warfare—the design of a sensor network of distributed autonomous sensing nodes.

MULTIOBJECTIVE PARETO-OPTIMAL DESIGN

Multiobjective Pareto-optimal design is the process of determining a design (or set of designs) that optimizes a set of objectives. The design parameters for all of the available designs are selected from the same set, and a specific combination of the design parameters defines a unique design. Related to but different from this is the notion of multidisciplinary design optimization, in which conflicting engineering disciplines attempt to achieve a common optimization goal by using differing sets of parameters. In the multiobjective Pareto-optimal design framework, the notion of multiple objectives is explicit, so that the relationships between objectives are only available after the optimization is complete. This is in contrast to non-Pareto approaches to multiobjective optimal design, in which the selection of a unique objective is made prior to the determination of optimal designs. Since the multiobjective Pareto-optimal design approach yields tradeoffs between designs, it is also beneficial in areas of decision support. In such cases, decision-makers are not only faced with conflicts, but the consideration of options (designs) is complex because the options affect the different objectives in different manners. The term "design" is thus used loosely here to mean a physical design, an action choice, or a set of parameters affecting the outcome of some event. All of these possible definitions are covered by the same optimization methodology.

The development of multiobjective Pareto-optimal design techniques follows two main branches: one in the computer science community and one in the mathematics community. In the computer science community, the methods are based largely on evolutionary computation strategies (see Deb[4] and Coello Coello[5] for summaries). Evolutionary methods are computing techniques that evolve designs according to rules that mimic evolutionary processes found in nature. One of the most common of these methods in single-objective design optimization is the genetic algorithm.[6] In general, the evolutionary approaches are based on forming large sets of designs (each set is called a generation) and comparing the designs within each generation. Features of the "better" designs in a generation are combined and used to create new designs for the next generation. This process iterates until convergence is achieved. In a single-objective setting, the convergence is to a single optimal design point; in a multiobjective setting, the convergence is to a design tradeoff space. The desirable characteristic of genetic approaches, particularly for complex problems, is that they require no numerical evaluation of the specific relationship between design parameters and resulting objective values. The disadvantages of these approaches are twofold: (1) an often sporadic approach to convergence and (2) the need to quantify what constitutes "better" designs. The first problem is a necessary drawback of such approaches and is compensated for only slightly in the more sophisticated genetic algorithms. The second problem is one that is often catastrophic in Pareto optimization, since the definition of better design in a multiobjective setting is not connected directly to the specific values of any objectives. For instance, a design may not be rated very highly with respect to either of two objectives but may fill in an important gap in the Pareto tradeoff surface. For engineering design and decision support, a multiobjective optimization method must address this drawback. Comparative studies of different evolutionary approaches to multiobjective optimal design are given by Van Veldhuizen,[7] Van Veldhuizen and Lamont,[8] and Zitzler and Thiele.[9]

The mathematics community's approach to multiobjective Pareto-optimal design is based on the functional representation of objectives in terms of design parameters. Methods are based on the numerical optimization of combinations of those functions, and resulting designs are presented to the designer or decision-maker for interpretation. Miettinen[3] provides a comprehensive summary of these methods and also provides a useful classification of the approaches into four categories: no-preference methods, interactive methods, *a priori* methods, and *a posteriori* methods. In no-preference methods, any solution that is optimal in any of the objectives is acceptable; however, these methods are inappropriate for developing tradeoffs. Interactive methods require the constant interaction of a knowledgeable expert to adjudicate between designs and are thus inappropriate when multiple disciplines are involved. *A priori* methods assume prior knowledge of the relative importance of the objectives. Such knowledge effectively creates a single-objective problem, albeit one with an objective that is some (usually complicated) combination of other objectives. In *a posteriori* methods, the optimization problem is posed as the specific derivation of a tradeoff surface. This is the general design goal here; thus, *a posteriori* methods are the methods of interest in complex systems design and decision support.

The *a posteriori* methods generally involve repeatedly forming different combinations of the objectives in which the optimization of each objective provides a unique point on the tradeoff surface. The most commonly used of these methods is that of weighted objectives, in which different linear combinations of the objectives are formed to provide combined objectives. However, as Das and Dennis[10] point out, such methods often perform poorly in providing a useful spread of points throughout the tradeoff surface. Normal boundary intersection[11] is an approach that overcomes this burden, but this comes at the cost of very complicated objective combinations that are usually poorly numerically conditioned. The reliance on gradient schemes to find the optimal value of the combined objective is the usual difficulty encountered with these approaches. The goal here is to build on these *a posteriori* methods while using the evolutionary approaches to replace the numerical optimization component and thereby overcome the current difficulties. The approach developed here is thus a combination of the mathematical approach (using the *a posteriori* method of normal boundary intersection) and the computer science approach (using genetic algorithms).

MATHEMATICAL DESCRIPTION OF PARETO OPTIMIZATION

The development of a design tradeoff surface is represented mathematically by analyzing conflicting objectives. In particular, suppose a design is dependent on a set of parameters, represented by the vector $X \in R^k$, so that the j-th design is uniquely described by specifying the parameter set $X = X_j$. Throughout this development, scalar values are represented by lowercase letters (such as x), vector values are represented by uppercase letters (such as Y), and matrix values are represented by uppercase bold letters (such as $\mathbf{Z}$). Also, all vectors are assumed to be column vectors, by default, with a row vector represented as the transpose of a column vector. The designer's primary task is to find a design that optimizes a set of n objectives. These objectives may represent things such as weight, cost, strength, detection performance, deployability, and combat effectiveness. Each objective is numerically represented by a function $f_i : R^k \to R^1$ of the parameters X, such that $f_i(X_j)$ represents the i-th objective evaluated for the j-th design. Without any loss of generality, the assumed goal of all objectives is to obtain the minimum of $f_i(X)$. For objectives that are naturally represented as maximizing functions, a simple mapping of $f_i(X) \mapsto -f_i(X)$ converts the goal to one of minimization.

Consider a class D of achievable designs X_j. Given a set of n minimization objectives, equation (1) represents the general multiobjective design optimization problem as

$$\min_{X \in D} \quad F(X) = \begin{bmatrix} f_1(X) \\ f_2(X) \\ \vdots \\ f_n(X) \end{bmatrix}. \tag{1}$$

With the design space X represented by a set of numerical parameters, the class of achievable designs D is represented by a set of equality, inequality, and bound constraints as follows:

$$D = \{X : H(X) = 0, \quad G(X) \leq 0, A \leq X \leq B\}. \tag{2}$$

For some simple problems, the optimal solution of $F(X)$ exists uniquely because the objectives are not in conflict. That is, the optimization of one objective presupposes the optimization of the other $(n - 1)$ objectives, and the problem is effectively one of single-objective optimization. In such cases, the problem is solved by performing careful engineering analyses *a priori* to determine the single objective to optimize. While such solutions are desirable, they are usually not attainable in real, complex systems. The objectives are often in conflict, and the optimization of one objective implies suboptimal values of the other objectives. A simple example of this kind of conflict is in performing cost–benefit optimization, where minimizing the cost is in conflict with maximizing the benefit. These problems require explicit multiobjective optimization techniques in order to study the set of design tradeoffs that separate the design solutions X that minimize the individual objectives $\{f_i\}$.

To study the design tradeoffs found in combinations of suboptimal solutions, the dominance of designs (or lack thereof) must be formally determined. In this context, the dominance relationship between two vectors is defined by the following statement:

- The objective vector $F(X_j)$ dominates the vector $F(X_i)$ if, and only if, $f_m(X_j) \leq f_m(X_i)$ for all $m = 1, \ldots, n$, and $f_m(X_j) < f_m(X_i)$ for some m.

The dominance relationship is written as $F(X_j) \prec F(X_i)$. This mathematical notion of dominance is very intuitive. If a design is better than another design in at least one objective and no worse in any objective, then it is a better design and hence dominates the other design. Given that there may be many solutions that dominate (in the context of the multiobjective optimization vector F), it is impossible to choose between two different dominant designs in the absence of further information that is beyond the purview of the statement of the optimization problem.

Given the notion of dominance of designs, the concept of Pareto optimality can be defined. The economist Pareto[12] was the first to mathematically formalize the notion of multiobjective tradeoff space exploration. Designs are defined as Pareto-optimal if they meet one of the two following criteria:

- Criterion 1: Globally Pareto-Optimal — If there is not an $X \in D$ such that $F(X) \prec F(X^*)$ for some $X^* \in D$, then X^* is a globally Pareto-optimal design.

- Criterion 2: Locally Pareto-Optimal — Let $B(X)$ represent the k-dimensional ε-neighborhood about a design X. If there is not an $X \in B(X^*)$ such that $F(X) \prec F(X^*)$, then X^* is a locally Pareto-optimal design.

The set of Pareto-optimal designs is defined as the union of the sets of globally Pareto-optimal solutions and locally Pareto-optimal designs. Because of these criteria, Pareto-optimal designs are also referred to in the optimization literature as nondominated designs.

As an example of Pareto dominance, consider a two-dimensional parameter space $X = (x_1, x_2)$ with the design space D defined by the constraint

$$D = \{(x_1, x_2) : -5 \leq x_1 \leq 10, -5 \leq x_2 \leq 10\} . \qquad (3)$$

Within this constraint, the goal is to minimize the objective vector $F(X)$, given by

$$F(X) = \begin{bmatrix} x_1^2 + x_2^2 \\ (x_1 - 5)^2 + (x_2 - 5)^2 \end{bmatrix}. \tag{4}$$

Figure 1 shows the set of all designs $X \in D$ mapped into objective space as the shaded region. The solid red line represents the Pareto-optimal set of designs. Note that, in the objective space (f_1, f_2), the Pareto-optimal set is on the boundary of the set of achievable designs. This is generally true for all continuous problems. However, note also that the Pareto-optimal set is not given by the entire boundary of the achievable set, since other boundary designs are dominated by designs that are not on the boundary (i.e., designs that are better in both objectives). Thus, a plausible strategy for Pareto optimization may be to find the boundary and then eliminate points that are dominated. However, even in this simple example, that proves difficult because of the complexity of the mapping from parameter space (x_1, x_2) to objective space (f_1, f_2). This becomes even more of a problem when the objective values are only available via expensive simulations, as is often the case in complex engineered systems. Thus, the goal of Pareto optimization methods is to provide a reasonable estimate of the set of Pareto solutions (or Pareto front) in as efficient a manner as possible (i.e., without generating the entire space of achievable designs and analyzing the boundary).

This problem illustrates an important simplification when it is examined in parameter space (the space of points $X \in D$). The first objective, $f_1 = x_1^2 + x_2^2$, is a measure of the Euclidean distance from the design point $X = (x_1, x_2)$ to the origin $X = (0, 0)$; the second objective, $f_2 = (x_1 - 5)^2 + (x_2 - 5)^2$, is a measure of the Euclidean distance from the design point $X = (x_1, x_2)$ to the point $X = (5, 5)$. Drawing these two critical points in the parameter space, as shown in figure 2 by the plus signs, one can interpret the multiobjective Pareto optimization problem as follows. Consider a sample design point Z, marked by the asterisk (*) in figure 2. The circle that goes through point Z represents all the designs that have the same value of objective f_1 as Z (i.e., they are all the same distance from $X = (0, 0)$). Obviously, point Y (marked with a small circle (o)) represents the point on the larger circle that has the minimum distance to

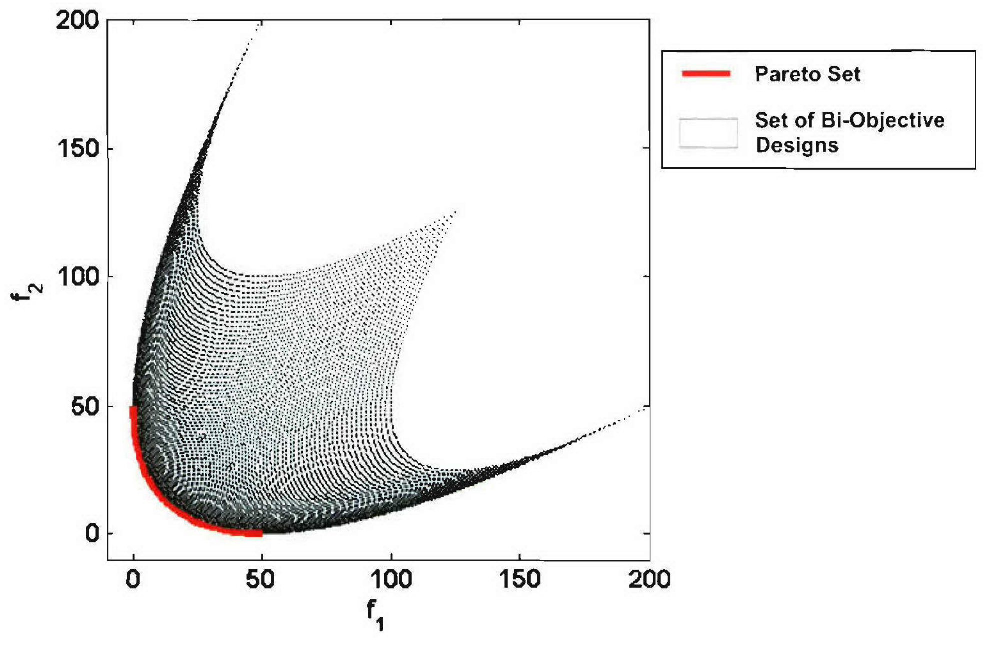

Figure 1. Example of a Set of Bi-Objective Designs Under a Design Constraint D Compared with the Set of Pareto Solutions

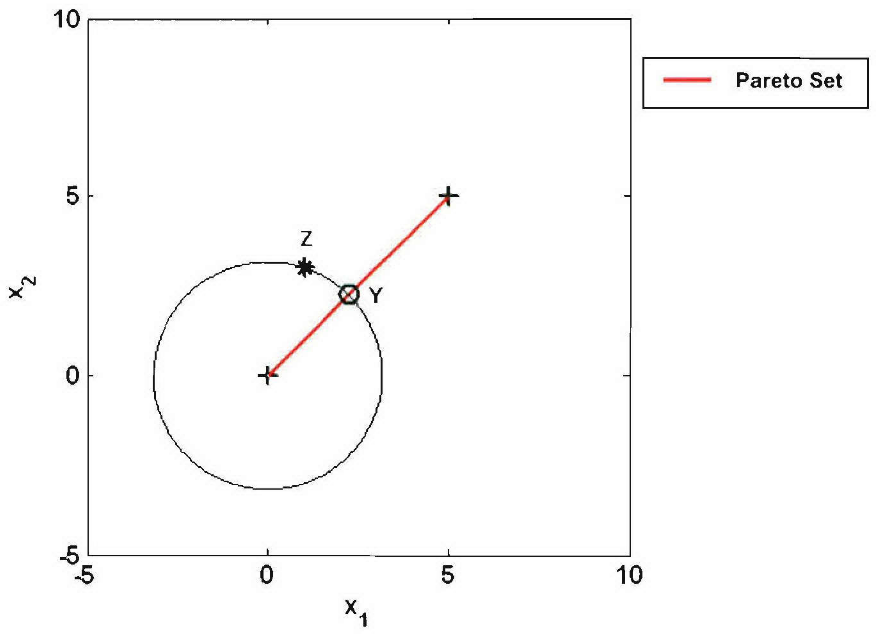

Figure 2. View of the Parameter Space of the Pareto-Optimal Set in Figure 1

point $X = (5, 5)$ and, thus, $Y \prec Z$. In fact, point Y dominates every other point on the circle, thus providing a Pareto-optimal design. Continuing this analysis for circles of arbitrary size shows that the set of Pareto-optimal designs are represented by the line segment between points $X = (0, 0)$ and $X = (5, 5)$, as shown by the red line in figure 2. Note that the same result is obtained with a similar construction using circles around point $X = (5, 5)$. This Pareto-optimal set is intuitive: any point that is not on the line segment is dominated by the nearest point to it that resides on the line segment. The mapping of these points to objective space provides the same Pareto-optimal set of designs as shown in figure 1. Thus, the concept of Pareto optimality is consistent with analysis in the objective space or in the parameter space. This example illustrates that simple derivations of Pareto-optimal sets are found by switching between objective and parameter space; however, such "gimmicks" are applicable only in the simplest of analytical cases and are rarely found in complex system designs where parameter spaces are very large.

EXISTING PARETO OPTIMIZATION APPROACHES

As mentioned above, the methods for numerically determining the approximation to the set of Pareto-optimal designs fall into two primary categories: optimization of objective functions via mathematical gradient methods and approximation via evolutionary methods. The former of these methods is based on knowing the functional relationships between the design parameters X and the objectives F. If such relationships are not available analytically, they must be determined empirically for such methods to work. Often, such things as gradients of the objective with respect to the design parameters are required, and the empirical evaluations must therefore be made on a fine enough scale to create the required accuracy in those calculations. The evolutionary methods have no such requirement to determine the functional relationship because they are based entirely on the manipulation of sets of points representing designs in the objective space. This generally makes evolutionary methods much more practical in complex systems design. In the following sections, the algorithmic approaches to each of the most popular methods in each category are described. The discussion is limited to *a posteriori* methods (in the nomenclature of Miettinen[3] as described previously) in which the optimization goal is to develop an approximation of the set of Pareto solutions to allow an informed decision-maker to make better design choices.

Gradient Solution Methods

The methods based on functional relationships between design parameters and objectives are based on gradient searches in the objective space; thus, they are referred to as gradient solution methods. In a single-objective problem, this is the simple hill-descent problem that is studied in elementary calculus. Unfortunately, many of the problems encountered in single-objective problems (convergence to local minima, poor estimates of gradient values, etc.) become even more troublesome when the size of the parameter space (the dimension of X) increases, as is often the case in complex systems design. These problems are further exacerbated when the size of the objective space (the dimension of F) is increased to include multiple objectives. As a practical matter, all gradient solution methods must eventually rely on some notion of dominance, and, thus, in effect reduce to a single objective. However, by considering the problems as multiobjective design optimizations rather than immediately thrusting them into a single-objective setting, the important design tradeoff considerations that must be made by a conscientious decision-maker can be identified. With multiple objectives, the gradient solution is often performed under a combined objective or by converting an objective to a constraint. The main techniques based on each of these approaches are briefly described here.

Perhaps the simplest (and, thus, the most commonly applied) method to solve a multiobjective problem is to convert the multiple objectives to a single objective by performing a weighted sum of the individual objectives. This technique is very common because of the ease of implementation. The technique consists of simply forming the net objective $f_{WS}(X)$, given by

$$f_{WS}(X) = \alpha_1 f_1(X) + \alpha_2 f_2(X) + \cdots + \alpha_n f_n(X). \tag{5}$$

The only other requirement for implementation is a choice of weights $\{\alpha_j\}_{j=1}^{n}$. The user then solves the single-objective problem as follows:

$$\min_{X \in D} f_{WS}(X). \tag{6}$$

One disadvantage of this method is that it provides only a single point of the set of Pareto solutions. This complication is usually avoided by running the problem repeatedly for different sets of weights. However, the *a priori* choice of many sets of weights does not often lead to good spreads of solutions along the Pareto set. Thus, many features of the Pareto set, such as sharp corners or concave sections, are often missed by this approach, as was pointed out by Das and Dennis.[10] Furthermore, the convergence properties of the combined objective $f_{WS}(X)$ are often worse than those of the individual objectives. For these reasons, this method is used only for the simplest of problems and is therefore inappropriate for generating approximations to the Pareto set for complex systems.

The other primary gradient solution method is referred to in the literature as the epsilon constraint method.[3] In a manner similar to the weighted sums method, the epsilon constraint method relies on performing a series of single-objective optimizations, each one leading to a single point on the approximation to the Pareto set. However, instead of combining objectives, the epsilon constraint method converts all of the objectives except for one to constraints and then modifies that constraint value for each of a series of single-objective problems. Mathematically, the method consists of modifying the constraint space D from equation (2) to D_ε, as

$$D_\varepsilon = \{X : H(X) = 0,\, G(X) \le 0,\, A \le X \le B,\, f_j(X) \le \varepsilon_j, j = 2, \ldots, n\}, \tag{7}$$

and then solving the single objective optimization problem as

$$\min_{X \in D_\varepsilon} f_1(X). \tag{8}$$

The choice of constraint values $\{\varepsilon_j\}_{j=2}^n$ changes for each iteration, and each choice leads to a new point on the approximate Pareto set. This method suffers from some of the same problems as the weighted sums method: namely, the method does not lead to a good spread of solutions along the set of Pareto solutions, and concave sections of the Pareto set are often missed. The method does, however, find sharp corners in the Pareto set, and the convergence is often very similar to that of optimizing objective $f_1(X)$ under the original constraint. One additional drawback to this method is that the portion of the approximate Pareto set that is generated is often highly dependent on the choice of which objective is labeled $f_1(X)$ (the one that is left out of the constraint space). Finally, as with all gradient solution methods, the gradients needed in the resulting single-objective problem require repeated k-dimensional derivative calculations (for $X \in R^k$). This requirement makes these methods impractical for complex design problems that have large-dimension parameter sets.

Evolutionary Solution Methods

Evolutionary methods of optimization are based on forming large sets of potential designs X, evaluating the relative performance of the designs within each set, and using some evolutionary principle to combine traits of members of each set to generate a new set of candidate designs. This process is repeated until the iterations stabilize, at which time the solution is presumed to have converged to an (at least local) optimum. These evolutionary methods include techniques such as genetic algorithms, simulated annealing, particle swarm methods, and ant colony optimization. Since the methods only use objectives as a means for scoring individuals for "good" traits, they are more easily applied in a multiobjective setting than in gradient solution methods. Furthermore, nothing prohibits using evolutionary methods to solve the single-objective problems formulated when using techniques such as gradient search and epsilon constraint. However, such an approach loses one of the primary benefits of evolutionary methods—i.e., since each iteration involves many designs, the end of an iteration can lead to an approximation of the entire Pareto set, rather than a single point on the Pareto set. The discussion in this report is restricted to genetic algorithms because, relative to the other evolutionary approaches, they are easy to implement and can be readily applied to many diverse design problems.

Genetic algorithms attempt to imitate the process of natural selection by forming new designs based on random combinations of features from pairs of different good designs. The term genetic is used to represent this similarity between the optimization strategy and the biological process. Over many iterations (or generations in a biological context), the best features survive in designs that have good (or even optimal) features, without any mathematical description of the relationships between features (design parameters) and performance (design objectives). The basics of genetic algorithms are covered in many texts, such as the one by Goldberg;[6] the interested reader is referred to those texts for the details of implementation. In a standard genetic algorithm, the iteration process begins with a fixed set of designs (called a population) that is manipulated to create the next population. The iteration of populations continues until some state of convergence is reached or, more realistically, for a fixed number of iterations that is believed to be large enough for convergence. Individual designs in the population are represented by binary strings (i.e., X_1 = [0010001110]). For designs described by multiple parameters, subsets of the string may represent values of the individual parameters, although this is not a necessity. The algorithm simply requires that a unique string be used to specify a unique design. The size of these strings is connected to the amount of design freedom available; thus, a string length of ℓ typically relates to a possible set of 2^ℓ unique designs in constraint space D. The genetic algorithm assigns each design $X \in D$ a fitness $\phi(X)$ based on its objective values. This fitness relates to the goodness of the design; better designs have larger fitness values. How these fitness values are assigned is unique to each specific genetic algorithm. For either single-objective or multiobjective optimization, each design X is assigned a scalar fitness $\phi(X)$. This feature makes genetic algorithms particularly attractive for multiobjective problems. Using the set of fitness values, designs are chosen at random (not uniformly, but weighted by their fitness, so that high-fitness designs are chosen more often) and formed into pairs for recombination (or reproduction in the biological context). The two strings representing a pair of recombining designs are then combined into two new strings, and the new corresponding designs (each new string corresponds to a new design) are placed into the next population. Once that population is full, the iteration is repeated until the process converges (if it is possible) in an optimal design or a Pareto-optimal set of designs. The details of recombination are also specific to each algorithm, although most use a simple splitting of traits. In that method, for two recombining designs X_1 and X_2 with string length ℓ, one new design takes the first m components of X_1 and the last $(\ell - m)$ components of X_2; the other new design takes the first m components of X_2 and the last $(\ell - m)$ components of X_1. The parameter m in this method is chosen at random for each recombination. Processes such as mutation provide further diversity by switching each bit in the string randomly (from 0 to 1 or from 1 to 0) with mutation probability μ. In practice, mutation probabilities of $\mu = 1/\ell$ are usually used. In the following paragraphs, the most common genetic approaches to multiobjective optimization are discussed to illustrate some of the negative features that the new method strives to eliminate.

The Vector Evaluated Genetic Algorithm (VEGA) may be viewed as the simplest method of multiobjective optimization. VEGA, developed in a doctoral thesis by Schaffer,[13] was one of the first genetic multiobjective optimization algorithms. Within each iteration of VEGA, the algorithm randomly divides the population of designs into the same number of groups as there are objectives, and each group is assigned an objective that all of its individual designs are evaluated against. For instance, for a four-objective problem, one quarter of the designs use objective $f_1(X)$; one quarter use $f_2(X)$; one quarter use $f_3(X)$; and the remaining quarter use $f_4(X)$. The fitness for recombination $\phi(X)$ is now given by $\phi(X) = 1 - f_j(X)$, where f_j is the appropriate objective. By this method, the designs that best comply with their assigned objective are given a greater chance of being used in the recombination process to create new designs. A disadvantage of VEGA is that solutions often cluster and migrate toward a single objective—converging to a single solution on the Pareto set instead of providing a sampling across the entire Pareto set. Random mutation of individual designs is often used to maintain the diversity needed to provide a spread of points across the Pareto set in this method. VEGA also tends to be sensitive to scaling issues between individual objectives. The method will often force the approximate Pareto set to a particular objective if the objectives are not weighted appropriately. Although these issues are independent of the size of the parameter set used in the individual designs, the performance of VEGA relies on the *a priori* scaling of objectives to make the algorithm practical in complex systems design optimization.

The Niched Pareto Genetic Algorithm (NPGA) was developed by Horn et al.[14] to eliminate the problem of convergence to a single point on the Pareto surface that often occurred with implementation of methods like VEGA. NPGA avoids the problem by limiting the connection between the numerical values of the objective values $F(X)$ and the numerical value of the fitness for recombination $\phi(X)$. At each iteration, the NPGA method takes two individual designs (labeled X_1 and X_2) from the population of designs and compares them to a subpopulation of designs (D_{SUB}) to test for domination. If $F(Y) \prec F(X_1)$ for some $Y \in D_{SUB}$, and there is no $Z \in D_{SUB}$ such that $F(Z) \prec F(X_2)$, then X_2 is selected as a survivor (since X_1 is dominated by D_{SUB} and X_2 is not). Similarly, X_1 is selected as a survivor if X_2 is dominated and X_1 is not. If they are both dominated (i.e., $F(Y) \prec F(X_1)$ for some $Y \in D_{SUB}$, and $F(Z) \prec F(X_2)$ for some $Z \in D_{SUB}$) or if neither are dominated (i.e., there is no $Y \in D_{SUB}$ such that $F(Y) \prec F(X_1)$, and there is no $Z \in D_{SUB}$ such that $F(Z) \prec F(X_2)$), then a niching computation is used to choose the survivor. The niching algorithm examines the number of designs in the population whose Euclidean distance to each design point is less than a fixed number σ. The design with the lower value of σ (and hence the more unique of the two) is chosen as the survivor. This process is repeated with every pair of designs in the population. All surviving designs are then assigned equivalent fitness (i.e., $\phi(X) = 1$ for survivors and $\phi(X) = 0$ for nonsurvivors) in the recombination process. The convergence properties of NPGA are much more sporadic than for other multiobjective genetic algorithms. Also, mutation often needs to occur more frequently than in other techniques; without large mutations, the algorithm tends to converge to small clusters of designs that lie on the set of Pareto solutions. The use of niching tends to make this method very dependent on the scaling between objectives (since the niching distance is computed as a Euclidean distance in the n-dimensional objective space). However, when the objective scaling and mutation parameters are carefully chosen, the method provides accurate representations of Pareto sets. The scaling problems make this method particularly unsuitable for complex systems design applications.

The Nondominated Sorting Genetic Algorithm (NSGA) by Srinivas and Deb[15] and its derivative methods[16, 4] are currently the most successful of the available multiobjective genetic techniques. NSGA sorts the individual designs in a population into fronts based on their degree of dominance. Then, each front is assigned a fitness value, and the entire population is sampled for recombination. The sorting of designs into fronts is accomplished as follows. Given a populations of designs D_{POP}, the subset of designs D_1 that make up the first front is obtained by taking all nondominated designs, i.e., $X_j \in D_1$ if and only if there is no $Y \in D_{POP}$ such that $F(Y) \prec F(X_j)$. All of the designs in this front are assigned fitness values of $\phi(X_j : X_j \in D_1) = \phi_{MAX}$. The second front D_2 is formed by considering those individuals that are nondominated after the first front has been removed. That is, $X_j \in D_2$ if and only if there is no $Y \in D_{POP} \setminus D_1$ such that $F(Y) \prec F(X_j)$. All of the designs in the second front are assigned a slightly lower fitness value of $\phi(X_j : X_j \in D_2) = \phi_{MAX} - \varepsilon$. This process is repeated until all of the designs in the population have been assigned fitness values. Then, the population of designs is sampled for recombination. NSGA overcomes many of the problems of methods like VEGA and NPGA; in particular, it is the only popular method that does not rely on careful scaling between objectives and is thus well-suited for problems of complex systems design optimization. Suggested improvements to NSGA (such as NSGA-II,[16] which applies a niching process similar to NPGA) throw away this benefit to provide slight improvements in spread along the Pareto set. For that reason, those suggested improvements are not considered in this discussion of the application of NSGA to complex systems design.

The existing approaches to multiobjective Pareto-optimal design described above all have complications that make them unsuitable for use in complex systems design applications. In particular, the complexity of the relationship between parameters and objectives, the inability to handle large parameter spaces, the poor numerical scaling between the competing objectives, and the poor spread of points on the Pareto set that often occurs are all problems that must be overcome to have a good tool for complex systems design. The gradient solution methods have no way of overcoming the complexity of the relationships between parameters and objectives that are inherent in these applications, and they do not scale well with large parameter spaces. Therefore, the gradient solution methods are removed from further consideration. Although genetic algorithms avoid the numerical difficulties with complex objectives and work with reasonably large parameter spaces, they still have problems providing a good spread of design points along the Pareto set, especially when the multiple objectives are poorly scaled relative to one another. For example, cost, the time required to achieve success, endurance, power, and probability of mission success—factors that are often present in military systems design—all occur in very dissimilar units. To handle these problems, a new preprocessor for genetic multiobjective optimization solvers has been developed. This preprocessor provides good spread along the Pareto set in poorly-scaled multiobjective settings, while maintaining the desirable features of the existing genetic algorithm. The standard NSGA is used here as the baseline of the current state-of-the-art for application in complex systems design, and examples are provided in which the new preprocessor improves the performance of both the VEGA and NSGA methods.

GANBI METHOD

The use of normal boundary intersection techniques as a method to obtain an estimate of the Pareto front was first developed in a doctoral dissertation by Das[17] and later documented in a journal article by Das and Dennis.[11] However, the approach of Das and Dennis was limited to situations where the objective vector *F*(*X*) has differentiability properties that allow the use of gradient search and other nonlinear programming techniques. Complex engineered systems often have objective functions that are not readily represented by closed-form mathematical expressions. In these cases, the computed numerical gradients that are required by the nonlinear programming techniques are not readily available; when they are available, the computations are of questionable accuracy. The work of Das and Dennis is extended in this report to extract essential features of the normal boundary intersection technique that make it appropriate for use as a preprocessor for general-purpose multiobjective solvers. The result is the development of a new method to be applied as a preprocessor for multiobjective genetic algorithm solvers. The new method is referred to as the Genetic-Algorithm-Based Normal Boundary Intersection (GANBI) method.

MATHEMATICAL DERIVATION OF GANBI

Throughout the development of GANBI, the problem under consideration is the *n*-objective multiobjective optimization problem (*MOP*), for which the goal is to minimize the vector objective *F*(*X*) subject to the constrained design space *D*. *MOP* is defined by equation (9):

$$MOP: \begin{cases} \min\limits_{X \in D} F(X) = \begin{bmatrix} f_1(X) \\ f_2(X) \\ \vdots \\ f_n(X) \end{bmatrix}, \quad n \geq 2 \\ \text{subject to} \quad D = \{X : H(X) = 0, G(X) \leq 0, A \leq X \leq B\} \end{cases} \quad . \qquad (9)$$

The *individual minimum* of the i-th objective, f_i* is defined as

$$f_i* = f_i(X_i*), \tag{10}$$

where

$$X_i* = \{X \in D : f_i(X) \le f_i(Y), \forall\, Y \in D\}.$$

Simply put, this is the minimum value that the i-th objective takes over the entire span of designs in the constrained design space D, with no regard given to the existence or value of the other objectives.

Given the set of all n individual minima $\{f_i*\}_{i=1}^{n}$, the *convex hull of individual minima* (*CHIM*) is defined by equations (11) through (13).[11] Let $F*$ be the vector of individual minima given by

$$F* = [f_1*, f_2*, \ldots, f_n*]^T, \tag{11}$$

and let $F_i* = F(X_i*)$. Define $\mathbf{\Phi}$ as the $n \times n$ matrix whose i-th column is given by $F_i* - F*$, i.e.,

$$\mathbf{\Phi} = \begin{bmatrix} 0 & f_1(X_2*) - f_1(X_1*) & \cdots & f_1(X_n*) - f_1(X_1*) \\ f_2(X_1*) - f_2(X_2*) & 0 & \cdots & f_2(X_n*) - f_2(X_2*) \\ \vdots & \vdots & \ddots & \vdots \\ f_n(X_1*) - f_n(X_n*) & f_n(X_2*) - f_n(X_n*) & \cdots & 0 \end{bmatrix}. \tag{12}$$

Then, *CHIM* is the set of all the points that are convex combinations of the columns in equation (12); specifically,

$$CHIM = \{\mathbf{\Phi} B + F* : B = [b_1, b_2, \ldots, b_n]^T, \textstyle\sum b_i = 1, b_i \ge 0\}. \tag{13}$$

This set of points represents all of the possible convex combinations of the various individual minima. *CHIM* provides a very coarse approximation to the Pareto surface. In a bi-objective optimization problem, *CHIM* is simply the line segment that connects the two individual minima. In a tri-objective problem, *CHIM* is the triangular planar region that connects the three individual minima. The region defined by *CHIM* changes with the number of objectives in the problem.

The GANBI method is based on the concept of building up an approximate Pareto surface—beginning with *CHIM*—and attempts to "push out" this coarse approximation as far as possible at regularly spaced intervals while staying within the constraints of the design space *D*. To illustrate the Pareto approximation used by GANBI, consider the bi-objective problem from figure 1, which is referred to as problem MOP_1:

$$MOP_1: \begin{cases} \min\limits_{X\in D} F(X) = \begin{bmatrix} x_1^2 + x_2^2 \\ (x_1-5)^2 + (x_2-5)^2 \end{bmatrix} \\ \text{subject to} \quad D = \{(x_1, x_2): -5 \le x_1 \le 10, -5 \le x_2 \le 10\} \end{cases} . \tag{14}$$

For MOP_1, *CHIM* is formed by first defining the points that are individual minima, as given by

$$\begin{aligned} f_1{}^* &= \min_{X\in D}(x_1^2 + x_2^2) = 0, \\ f_2{}^* &= \min_{X\in D}((x_1-5)^2 + (x_2-5)^2) = 0. \end{aligned} \tag{15}$$

The points defined by $f_1{}^*$ and $f_2{}^*$ occur at design parameters $X_1{}^* = (0, 0)$ and $X_2{}^* = (5, 5)$, respectively. These specific parameter values are unimportant to the *CHIM* evaluation. Each design parameter is used to obtain the value of the corresponding nonoptimized objective, as given by

$$\begin{aligned} f_1(X_2{}^*) &= (x_1^2 + x_2^2)\big|_{X=(5,5)} = 50, \\ f_2(X_1{}^*) &= ((x_1-5)^2 + (x_2-5)^2)\big|_{X=(0,0)} = 50. \end{aligned} \tag{16}$$

Thus, *CHIM* is given by all points defined by equation (17):

$$F_{CHIM} = \begin{bmatrix} 0 & 50 \\ 50 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 50 - 50b_1 \\ 50b_1 \end{bmatrix}, \tag{17}$$

where b_1 takes values in the range [0, 1]. This yields the simple line segment in figure 3 (black line), where the asterisks (*) represent the individual minima. This is obviously a very coarse approximation to the Pareto surface (red line). However, if *CHIM* is imagined as represented by a "rubber band," the approximation is improved by "pushing" the band out from its center point as far as possible (within the constraints of maintaining objective values F that are achievable using parameters X, contained in the design space D). This level of approximation is represented in figure 4. The process improves if, instead of using only the center point, k evenly-spaced points are used along *CHIM*, as shown in figure 5 for the case of $k = 4$.

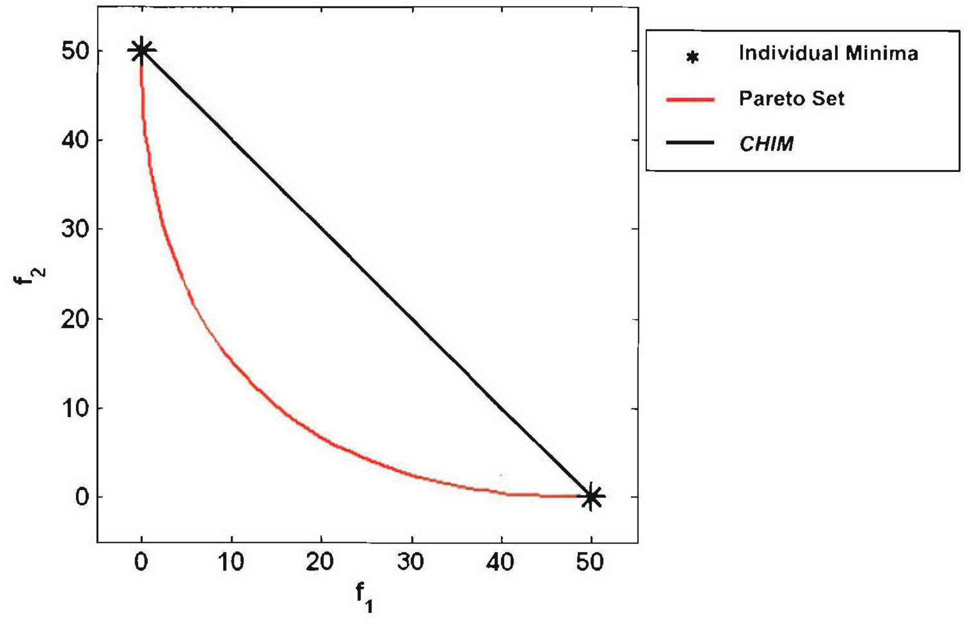

Figure 3. Individual Minima, Pareto Set, and CHIM for the Example in Figure 1

This process of pushing *CHIM* toward the Pareto surface is based on the geometrical idea that underlies all normal boundary intersection methods. The geometrical construction is identical for a general n-objective problem and can be imagined by replacing the rubber band with an $(n - 1)$-dimensional surface (e.g., a "rubber sheet" for a tri-objective problem).

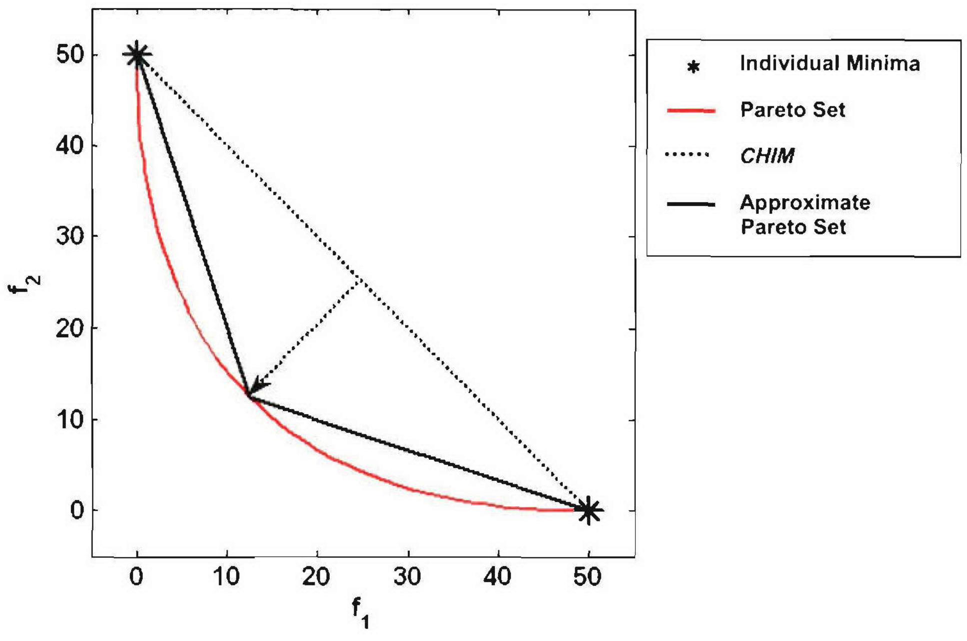

Figure 4. Approximate Pareto Set Found by Pushing Away from the Midpoint of CHIM from Figure 2

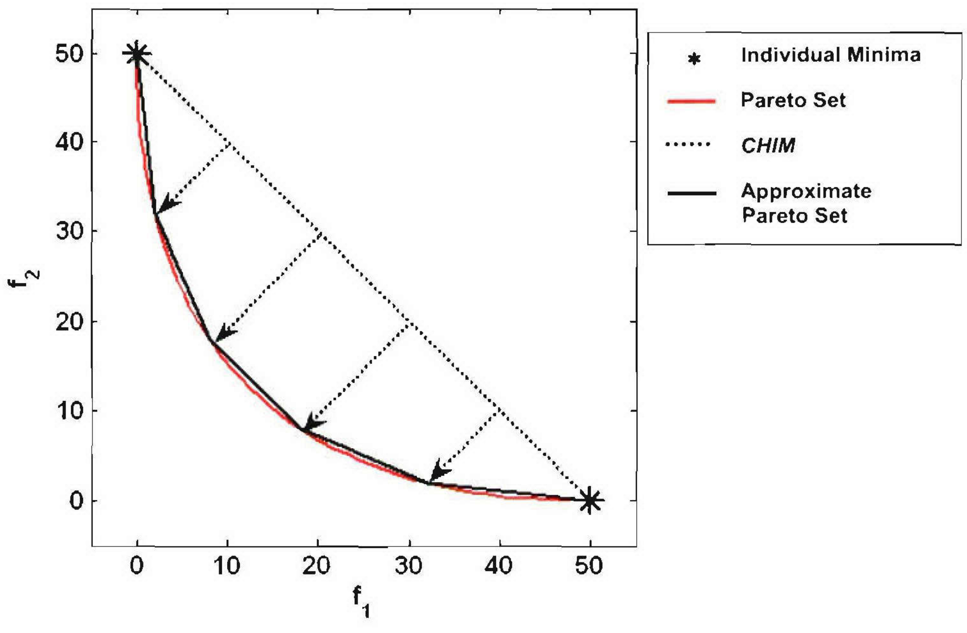

Figure 5. Approximate Pareto Set Found by Pushing Away from Four Points Along CHIM from Figure 2

The mathematical details of modifying *CHIM* to approximate the Pareto surface are required to write the details of the GANBI method. The development begins by considering the problem of describing k evenly-spaced points along *CHIM*, which are given by $F_i = \Phi B_i + F^*$, where

$$B_i = \begin{bmatrix} i/(k+1) \\ (k+1-i)/(k+1) \end{bmatrix}, \quad i = 1,\dots,k, \tag{18}$$

for the bi-objective case. For the general n-objective case, the formulation is less intuitive, but it can be shown that B_i can be created from a simple process. Assume that a set of values for b_1 is desired, given by $b_1 = \{0, \frac{1}{p}, \frac{2}{p}, \dots, \frac{p-1}{p}, 1\}$, where p is an integer (and $p > 1$). Now, the accepted values for the other b_j are given by

$$b_j = \left\{0, \frac{1}{p}, \frac{2}{p}, \dots, \left(1 - \sum_{i=1}^{j-1} b_i\right)\right\}, \quad \text{for} \quad j = 2, \dots, (n-1). \tag{19}$$

Once the combination of values for each different b_i is formed for $i = 1, \dots, (n-1)$, each combination yields the final value of

$$b_n = 1 - \sum_{i=1}^{n-1} b_i. \tag{20}$$

Finally, all of the points are removed that contain any of the individual minima, which are points where $b_i = 1$ for some i. This construction is consistent with the simple two-dimensional formula in equation (18). Also, the number of evenly-spaced points generated in this manner for an n-dimensional *CHIM* is given by k, where

$$k = \binom{p+n-1}{p} - n = \frac{(p+n-1)!}{p!\,(n-1)!} - n, \tag{21}$$

which then reduces to $k = p - 1$ for the bi-objective (n = 2) case. From the set of values $\{B_i\}_{i=1}^k$, $F_i = \Phi B_i + F^*$ is formed again. The set of points $\{F_i\}_{i=1}^k$ represent k evenly-spaced points along *CHIM*.

The GANBI method relies on pushing the approximate Pareto surface (given by *CHIM*) in the direction normal to *CHIM* at each of these k points. The vector normal to *CHIM* that goes through a given point F_i on *CHIM* is parametrized by $z \geq 0$; the vector is given by

$$N_i(z) = F_i - z\Phi^T U, \tag{22}$$

where U represents the vector of all ones, i.e., $U = [1, 1, \ldots, 1]^T$. The value of $z = 0$ represents the location of the normal at the point of intersection with *CHIM*, and values of larger z represent values pushing *CHIM* in the direction of the Pareto set. The goal in GANBI (as illustrated in figure 4) is to stay as close to possible to each of these k lines $\{N_i(z)\}_{i=1}^k$ while increasing z as much as possible. These objectives are obviously in conflict, and the multiobjective solution of this optimization problem provides an estimate of the Pareto surface that tends to lead to a much better spread of points along the Pareto set than solutions that do not employ the GANBI preprocessor.

The individual objective of moving as far along the line $N_i(z)$ as possible is, in itself, a multiple objective. One goal is to stay close to the line, and a second is to move far along it. Rather than consider these goals as further dimensions of the objective space, a simple joint objective is considered. Given a design point $F = F(X)$ for parameters X, the objective $h_i(F)$ is formed by

$$h_i(F) = d(N_i(z_{LS}), F) - 2z_{LS}, \tag{23}$$

where F is the given design point (in objective space), $d(A, B)$ is an appropriate measure of distance between points A and B (in objective space), and z_{LS} is an estimate of the value of z for the point on $N_i(z)$ that is closest to F. The factor of 2 before the z_{LS} term was included because, after experimentation with many test problems, it has been determined to facilitate convergence. The value of z_{LS} that minimizes the least-square (L_2) norm is used; the value is given by

$$z_{LS} = (A^T A)^{-1} A^T G_i(F), \tag{24}$$

where

$$A = \mathbf{\Phi}^T U, \tag{25}$$

and

$$G_i(F) = F^* - F - \mathbf{\Phi}^T B_i. \tag{26}$$

Note that B_i is the same vector used in the definition of the i-th point along *CHIM*. The distance measure used is the L_2 norm, which is given by

$$\begin{aligned} d(N_i(z_{LS}), F) &= \left\| G_i(F) - z_{LS} \mathbf{\Phi}^T U \right\|_2 \\ &= \left\| \left(I - A(A^T A)^{-1} A^T\right) G_i(F) \right\|_2, \end{aligned} \tag{27}$$

which leads to the final objective of

$$h_i(F) = \left\| \left(I - A(A^T A)^{-1} A^T\right) G_i(F) \right\|_2 - 2(A^T A)^{-1} A^T G_i(F). \tag{28}$$

The objective $h_i(F)$ is evaluated for each $i = 1, \ldots, k$ for each given design point F, and the resulting vector-valued objectives, $H(F) = [h_1(F), h_2(F), \ldots, h_k(F)]^T$, are the result of the GANBI preprocessor that is used in a standard genetic multiobjective solver.

The GANBI preprocessing method is summarized in the following six steps:

1. Find the vector of individual minima F^*.
2. Form the matrix $\mathbf{\Phi}$ as in the *CHIM* definition, and form the corresponding vector A.
3. Form k vectors B_i according to equations (19) and (20).
4. For each given design point F, perform steps 5 and 6.
5. Form $G_i(F)$ for each vector B_i.
6. Evaluate $h_i(F)$ according to equation (28).

The method is easily implemented.[*] Steps 1 through 3 are performed once at the beginning of a multiobjective design optimization; steps 4 through 6 are the only ones that need to be repeated at each iteration. Thus, the added computational complexity over that of standard multiobjective solvers is minimal and, in practice, it is usually compensated for by improved convergence.

NUMERICAL EXAMPLE OF GANBI

As an example of the evaluation of the GANBI preprocessor, consider again the problem MOP_1 from equation (14). As shown earlier, the vector of individual minima is given by $F^* = [0, 0]^T$, and the matrix $\mathbf{\Phi}$ is given by

$$\mathbf{\Phi} = \begin{bmatrix} 0 & 50 \\ 50 & 0 \end{bmatrix}, \tag{29}$$

which leads to a value for the vector A of $A = [50, 50]^T$. For an example set of $k = 4$ GANBI objectives, as shown in figure 4, B_1, B_2, B_3, and B_4 are given by

$$B_1 = \begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 0.4 \\ 0.6 \end{bmatrix}, \quad B_3 = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}, \quad B_4 = \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}. \tag{30}$$

[*]Numerical implementation of the GANBI method can be facilitated by contacting the author of this report.

The values of $G_i(F)$ are now given by

$$G_i(F) = \begin{bmatrix} -f_1 - 50b_{i2} \\ -f_2 - 50b_{i1} \end{bmatrix}, \tag{31}$$

where b_{ij} is the j-th component of vector B_i. The resulting objective function values are thus given by

$$h_i(F) = \sqrt{\frac{(f_1 - f_2 + 50b_{i2} - 50b_{i1})^2}{2} + \frac{(f_1 + f_2 + 50b_{i1} + 50b_{i2})}{50}}. \tag{32}$$

This formula can now be used to examine the set of three design points that yield the following objective values:

$$F(X_1) = \begin{bmatrix} 20 \\ 10 \end{bmatrix}, \quad F(X_2) = \begin{bmatrix} 40 \\ 20 \end{bmatrix}, \quad F(X_3) = \begin{bmatrix} 10 \\ 40 \end{bmatrix}. \tag{33}$$

The three design points (blue filled circles) are shown in objective space in figure 6, with the individual minima (∗), the Pareto set (red line), *CHIM* (dotted line), the approximate Pareto set (black line), and the set of normals $\{N_i(z)\}$ (dotted arrows). The values of the resulting GANBI objectives for the three design points are given by

$$H(F(X_1)) = \begin{bmatrix} 29.9 \\ 15.7 \\ 1.6 \\ 15.7 \end{bmatrix}, \quad H(F(X_2)) = \begin{bmatrix} 37.6 \\ 23.4 \\ 9.3 \\ 9.3 \end{bmatrix}, \quad H(F(X_3)) = \begin{bmatrix} 2.0 \\ 16.1 \\ 30.3 \\ 44.4 \end{bmatrix}. \tag{34}$$

From these results, it is clear that design X_1 is best matched to the third objective (and, thus, has its lowest objective value in the third component) and design X_3 is best matched to the first objective. Both of these values are very low compared with those for the other objectives because the designs sit almost directly on the corresponding normal lines to *CHIM*. It should be noted that, since X_1 is closer to the Pareto surface, the lowest objective value for X_1 (1.6) is lower than the lowest value for X_3 (2.0). In terms of matching its objectives, design X_2 is the worst

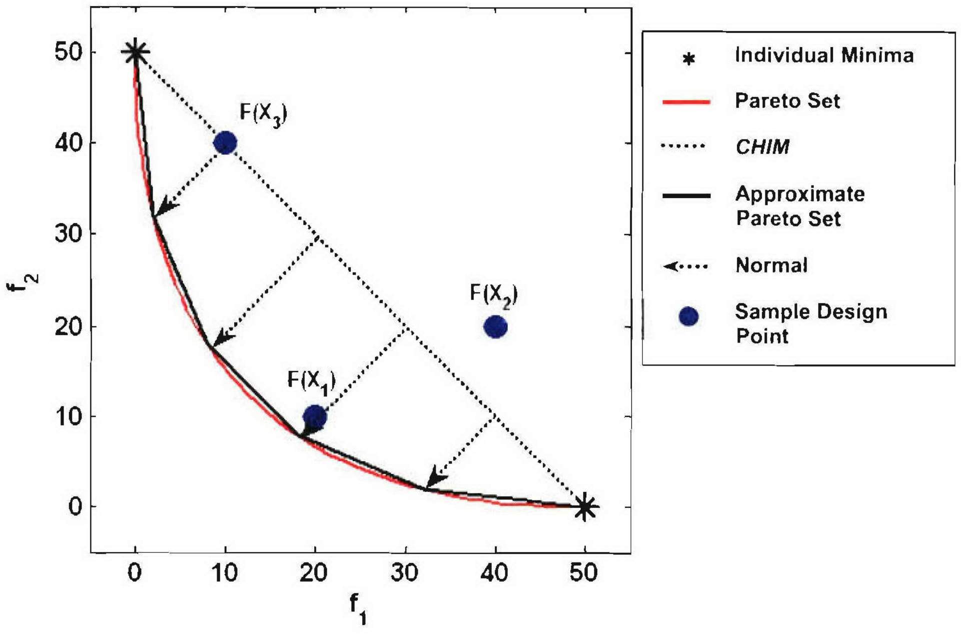

Figure 6. Sample Design Points Shown in Objective Space for Problem MOP_1

case. The lowest value for X_2 (9.3) is located midway between the third and fourth normals, so those two objectives are the the ones best matched by that design. However, the lowest value for X_2 is not nearly as low (i.e., not nearly as good a match with the corresponding objectives) as the low values for X_1 and X_3. The poorer match made by X_2 is demonstrated by two characteristics shown in figure 6: (1) the point is not near the two normal lines, and (2) the point is far from the direction of improving z values (in fact, it lies in the other direction from *CHIM*). Thus, the GANBI preprocessing provides a correct mathematical representation of the process that it is meant to perform.

Focusing again on problem MOP_1 from equation (14) allows an assessment of the numerical improvement obtained by using the GANBI preprocessor. The assessment uses $k = 4$ GANBI objectives on this $n = 2$ objective problem and shows all of the results in the original objective space. Recall that the total design space and the Pareto set are as shown in figure 1. First, the performance of the VEGA solver is compared with that of VEGA preprocessed with the GANBI method (referred to as GANBI/VEGA). Each design is represented by a 20-bit ($\ell = 20$) string, with the first 10 bits representing x_1 and the second 10 bits representing x_2. This provides a design space discretization of 1,024 × 1,024, yielding approximately 1.05×10^6 potential designs in D. A mutation probability of $\mu = 0.05$ is used to provide 40 designs at each iteration. Only 90% of the available designs are provided to the genetic algorithm at each iteration (the remaining 10% survive intact), which is a common practice used to speed final convergence. Figure 7 compares the performance of the GANBI/VEGA solver (blue circles) with that of the VEGA solver (green plus signs), both after 200 iterations. The true Pareto set is represented by the red line. Note that for 200 iterations of this problem, only 7,204 individual designs are ever evaluated, which leads to a sampling of only 0.069% of the design space D. Both approaches converge to solutions near the Pareto set. However, the VEGA solver used alone (without GANBI) gets many non-Pareto solutions near $f_1 = 60$ (these solutions are still borders of the achievable set, but they are dominated), while the GANBI/VEGA solutions tend to fall more heavily in the region of the Pareto set.

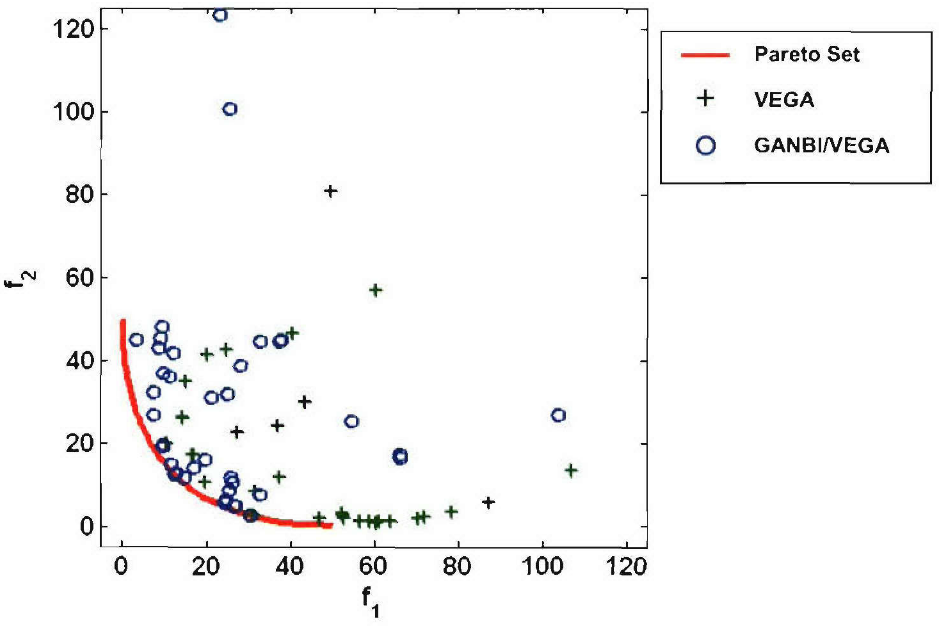

Figure 7. Comparison of Pareto Set and Approximate Pareto Sets Obtained Using VEGA and GANBI/VEGA Solutions for Problem MOP_I

Next, using the same problem, MOP_1, with the same basic genetic algorithm parameters, the NSGA solver is applied both with and without GANBI. The results after 200 iterations are shown in figure 8, where the green plus signs represent the NSGA sets for MOP_1. Once again, only 0.069% (approximately) of the design space has been sampled, yet both approaches yield approximations of the Pareto set. Both of the NSGA-based methods shown in figure 8 are closer to the Pareto set than the VEGA-based methods shown in figure 7. GANBI/NSGA did a better job than NSGA alone in providing a spread across the Pareto set. In particular, note that there are no NSGA solutions for the range of $20 \leq f_1 \leq 40$, yet there is a sampling of GANBI/NSGA solutions in that region of the Pareto set. From the examples illustrated in figures 7 and 8, it is clear that the GANBI method provides more spread along the Pareto set, with no degradation in closeness to the Pareto set, when used as a preprocessor for both VEGA and NSGA. Also, the best performance in these examples is obtained with the GANBI/NSGA combination. However, these results are for a simple analytical problem. The use of GANBI in a realistic complex systems design application is examined next.

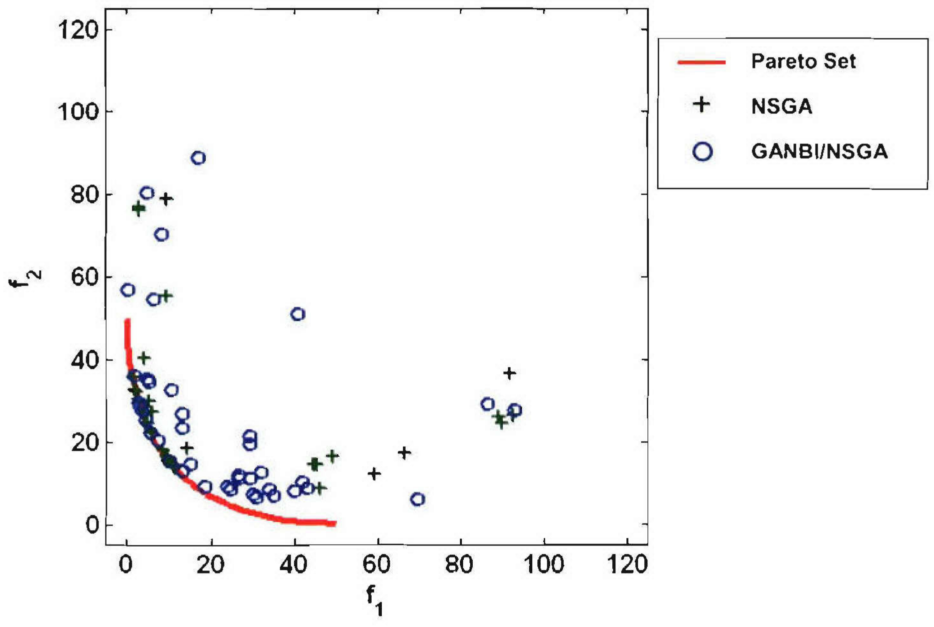

Figure 8. Comparison of Pareto Set and Approximate Pareto Sets Obtained Using NSGA and GANBI/NSGA Solutions for Problem MOP_1

USING GANBI FOR COMPLEX SENSOR NETWORK DESIGN

The development of sensor networks is a complex engineering challenge that involves the tradeoffs of many potentially conflicting design objectives. For a simple example, consider a set of m identical stationary sensors employed over a large spatial area to search for moving targets. This problem is typical of complex sensor network problems. A design goal of such a system may be to maximize detection performance, which can be done with the use of many sensors (i.e., make m very large). Unfortunately, such a system would have good detection performance but would also generate many false alarms. This leads to a second objective of minimizing the number of false alarms, which can be done with the use of very few sensors (i.e., make m very small). Obviously, these two objectives are in conflict. One solution to this conflict that is often heuristically stated is to increase the detection range r_d of the individual sensors. Although this solution theoretically leads to reducing the number of false alarms for a fixed level of detection performance (the false alarm rate per sensor remains constant, but fewer sensors are needed to cover the same area), it is often not practical because an increase in sensor detection range without an increase in sensor false alarm rate comes at an increased cost. Thus, there is a third design objective—constraining system cost—that is often not apparent in the initial statement of the design problem. The most straightforward method to resolve the conflict between detection performance and false alarm performance in a cost-effective manner is the use of track-before-detect strategies, in which multiple sensor detections must occur within the kinematic constraints of target motion.[18] Thus, the number of such detections γ is considered as an additional design degree of freedom, with $\gamma = 1$ corresponding to the traditional sensor coverage problem. Also, in the track-before-detect context, reference is made to the field-level functions of detection and false alarm as successful search and false search, respectively, because the track-before-detect process serves as a searching function beyond the performance of individual sensors.

The following specific example problem illustrates the effect of GANBI on complex sensor network design. Given a fixed search region $S \subset R^2$, determine the Pareto-optimal set of sensor field designs that maximizes the probability of successful search and minimizes the probability of false search under a constraint on total field cost. In this context, a sensor field design consists of a sensor detection range r_d, a number of sensors m (assumed to be uniformly distributed over the search area), and a number of detections γ to be used in a track-before-detect strategy. Thus, the optimization problem is stated as problem *DNS* as follows:

$$DNS:\begin{cases} \min\limits_{X\in D} F(X)=\begin{bmatrix}1-P_{SS}(r_d, m, \gamma)\\ P_{FS}(r_d, m, \gamma)\end{bmatrix} \\ \text{subject to } D=\{(r_d, m, \gamma): c(r_d, m)\le c_0, r_d \le r_{\max}, m\le m_{\max}, \gamma\le 4\}\end{cases}, \tag{35}$$

where the design space X is given by the triple of adjustable design parameters $X=(r_d, m, \gamma)^T$. The scope of the design space under consideration is based on limits of these values. The limiting values of $r_{\max}$ and $m_{\max}$ are chosen to provide coverage over the search area S, and the upper bound of $\gamma = 4$ is based on available tracking and fusion algorithms.

The functions P_{SS} and P_{FS} represent, respectively, the expected values of search effectiveness and the expected number of false searches that occur over an interval of time. For convenience, P_{SS} is expressed over the interval of time t_0 during which γ different sensors are required to detect the target (in track-before-detect) and P_{FS} is the probability of finding a kinematic sequence of false detections over time t_0 at least once during the period of 1 day. The use of 1 day as a P_{FS} time interval is a convenience to operators; it is readily replaced by any convenient time interval. The use of t_0 is a specific tactical parameter of interest that is part of the supporting command and control system for the sensor network. From the analysis of track-before-detect search in Wettergren,[18] the analytical expressions for P_{SS} and P_{FS} (over time t_0) are found by using equations (36) and (37), respectively:

$$P_{SS}=1-\exp(-p_d m\phi)\sum_{j=0}^{\gamma-1}\frac{(p_d m\phi)^j}{j!}, \tag{36}$$

and

$$P_{FS}=1-\exp(-\pi\, p_{fa} m)\left[\sum_{j=0}^{\gamma-1}\frac{(p_{fa} m\phi)^j}{j!}\right]^{\pi/\phi}, \tag{37}$$

where ϕ represents the fractional coverage of a single sensor extended by the expected target motion. This is given by

$$\phi = \frac{2r_d v t_0 + \pi r_d^2}{a_0} \tag{38}$$

for the case of a uniformly distributed homogeneous set of sensors distributed over a search region S of area a_0. The false search probability is further extended to the 1-day time interval (assuming time interval t_0 is much less than 1 day) as

$$P_{FS} = 1 - \left\{ 1 - \exp(-\pi\, p_{fa} m) \left[\sum_{j=0}^{\gamma-1} \frac{(p_{fa} m \phi)^j}{j!} \right]^{\pi/\phi} \right\}^{86400/t_0} . \tag{39}$$

Equations (36) and (39) are used to represent the objectives in the multiobjective optimization problem *DNS*. The cost function $c(r_d, m)$ is derived from combining the generic passive acoustics cost model from Traweek and Wettergren[19] with current costs of existing manufactured sonobuoys. All m sensor nodes are assumed to have the same cost, which leads to the following cost (in dollars):

$$c(r_d, m) = m \cdot (121 + 0.0061 \cdot r_d^2) . \tag{40}$$

Both the cost and search performance expressions make the two following critical assumptions about the sensors: (1) all sensors have "cookie-cutter" detection performance, in which the target is detected at all ranges in the interval $(0, r_d]$ with probability p_d and never detected beyond range r_d; and (2) all sensors have independent and identical probabilities of false alarm, given by a probability p_{fa} of reporting a false alarm over the time interval t_0.

The fixed parameters for any run of optimization problem *DNS* involve the target speed (v), the time for multiple detections to occur (t_0), the search region's area (a_0), the fixed maximum deployment cost (c_0), and the probabilities of detection (p_d) and false alarm (p_{fa}) for individual sensors. The specific values of these parameters used in the example are given in table 1. Note that the p_{fa} value shown in table 1 corresponds to a false alarm occurring over an integration time of 40 seconds with a probability of 10^{-3}. Limits are placed on the parameter values by using $r_{\max} = 10^4$ meters and $m_{\max} = 10^5$ sensor nodes for this size of search region in problem *DNS*.

Table 1. Values of Fixed Parameters Used for Example of Problem DNS

Parameter	Value
v	5 knots
t_0	30 minutes
a_0	100 nmi^2
c_0	\$1,000,000
p_d	0.9
p_{fa}	0.044

To implement problem *DNS* in GANBI, a sensor network design is represented by a string of length $\ell = 32$, with the first 15 bits corresponding to r_d, the next 15 bits corresponding to m, and the final 2 bits corresponding to γ. This provides 2^{15} possible values for both r_d and m, both of which are distributed exponentially across their acceptable domains (i.e., uniformly spaced in $\log_{10}(r_d)$ and $\log_{10}(m)$, respectively). The values of γ correspond to $\{1, 2, 3, 4\}$. At each iteration, 100 individual designs are provided, and 90% of these designs undergo genetic mating for the next iteration. Each new design is subjected to a mutation probability of $\mu = 1/32$, which corresponds to an expected value of one bit of mutation in each design. In the cases that were run, $k = 4$ GANBI objectives were created from the original two objectives of problem *DNS*. The cost constraint in the constraint space D is handled by a penalty method, in which each cost evaluation is checked against the constraint—the penalty threshold c_{wt} in this example. If the cost constraint is not violated, the objectives are passed to the solver. If the cost constraint is violated by more than c_{wt} = \$10,000, the objectives are set to unity (the maximal value) and sent to the solver. If the cost is violated by less than c_{wt}, the objectives are changed via $f(X) \mapsto 1-(1-(c-c_0)/c_{wt})(1-f(X))$ and sent to the solver. The value of c_{wt} used in this example was chosen by trial and error for the specific application. The specific form of the penalty function is limited to cases where $0 \le f(X) \le 1$, which is appropriate for problem *DNS*.

Figure 9 shows the results of applying VEGA with and without the GANBI preprocessor to multiobjective problem *DNS* through 200 iterations. Green plus signs represent the approximate Pareto sets obtained with the standard VEGA; the blue circles represent sets obtained with the GANBI/VEGA combination. The results are shown as P_{FS} versus P_{SS}, with the vertical scale in logarithmic form. The *DNS* objective is to move to the lower right corner of this plot; i.e., the objective is a sensor network that always performs search successfully and never has false searches. The cost constraint prohibits solutions from achieving this goal. Solutions in the upper right (near $P_{FS} = 1$ and $P_{SS} = 1$) are plentiful because they are easy to achieve (by employing many sensors with very small detection ranges and using $\gamma = 1$). It is clear from figure 9 that the GANBI preprocessor provides a much better spread of solutions along the entire Pareto front. Also, the GANBI/VEGA solutions generally dominate (are better in both objectives than) the solutions obtained with VEGA alone.

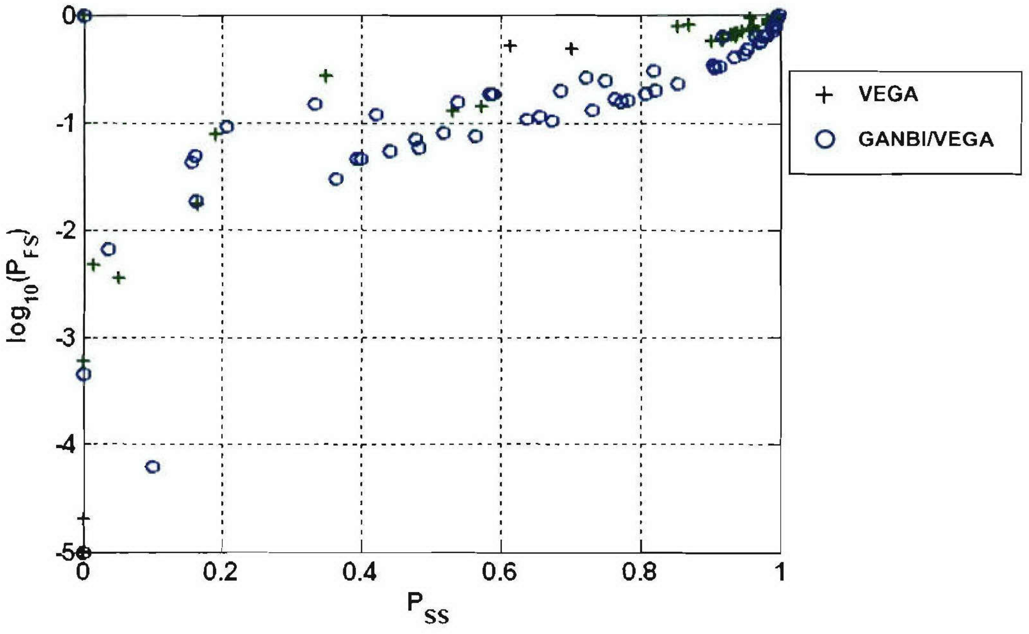

Figure 9. Comparison of Approximate Pareto Sets Obtained Using VEGA and GANBI/VEGA Solutions for Problem DNS

The experience with academic problems such as *MOP*$_1$ shows that there is a potential benefit in using GANBI with NSGA instead of VEGA. Figure 10 shows the results of applying NSGA with the GANBI preprocessor (blue circles) and without GANBI (green plus signs) to problem *DNS* through 200 generations. The scale is the same as that shown for figure 9. It is clear from the figure that, once again, using the GANBI preprocessor provides a much better spread of solutions along the entire Pareto front. Again, the solutions found with GANBI generally dominate (are better in both objectives than) those obtained with NSGA alone. It is also interesting to note that GANBI/VEGA and GANBI/NSGA obtain approximations to the Pareto front that are similar in quality, whereas VEGA and NSGA provide very different solution sets when run alone. This illustrates some of the robustness of the GANBI preprocessor; given enough iterations to converge, its performance is qualitatively independent of the specific underlying solver used. Also, note that each of these runs evaluated a total of 199 * 90 + 100 = 18,010 unique designs from a potential set of 2^{32} designs, which is only a $O(10^{-5})$ sampling of the design space. This shows that GANBI provides an effective sampling of complex design spaces for engineering complex systems.

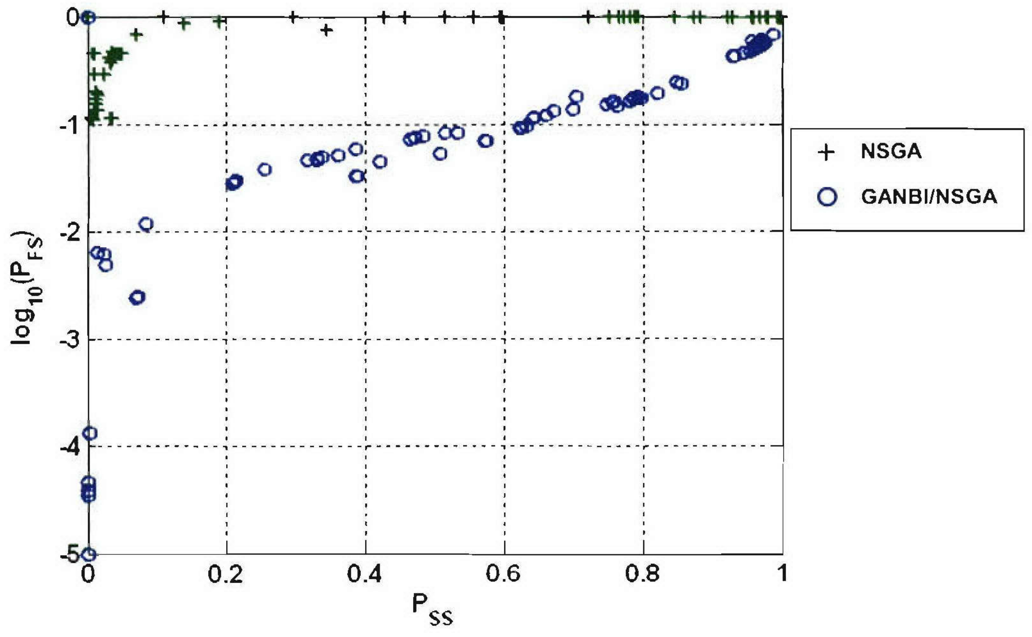

Figure 10. Comparison of Approximate Pareto Sets Obtained Using NSGA and GANBI/NSGA Solutions for Problem DNS

It is important to recall that a result point in the objective space plot corresponds to a system design (a specific value in the design space of $X = (r_d, m, \gamma)$). In figure 11, the GANBI/NSGA result from figure 10 is copied and marked with five sample results, chosen as a reasonable spread of samples along the approximate Pareto front. In table 2, the resulting values of the design parameters X are shown for each of these points, with the corresponding objective values. Note that, after performing the Pareto optimization, there appears to be a "best sensor range" around $r_d \approx 300$ meters for this application, and that range spans the Pareto curve. In fact, the curve seems to be parameterized by the number of sensors m, with the parameter γ adjusted for each design to guarantee the "best" choice in a Pareto set. This is especially convincing because the restriction on the sensor detection range in the design space D was less than 10^4 meters, with a sampling discretization of 2^{15} values within this range; yet the Pareto-optimal solution set seems to have converged very near to a single range value. Insights such as this can now be used to perform a detailed design. Performing the Pareto optimization before the detailed design stage gives the designer the advantages of fixing the optimum sensor range and knowing how the number of sensors used affects the tradeoff between search effectiveness and false searches.

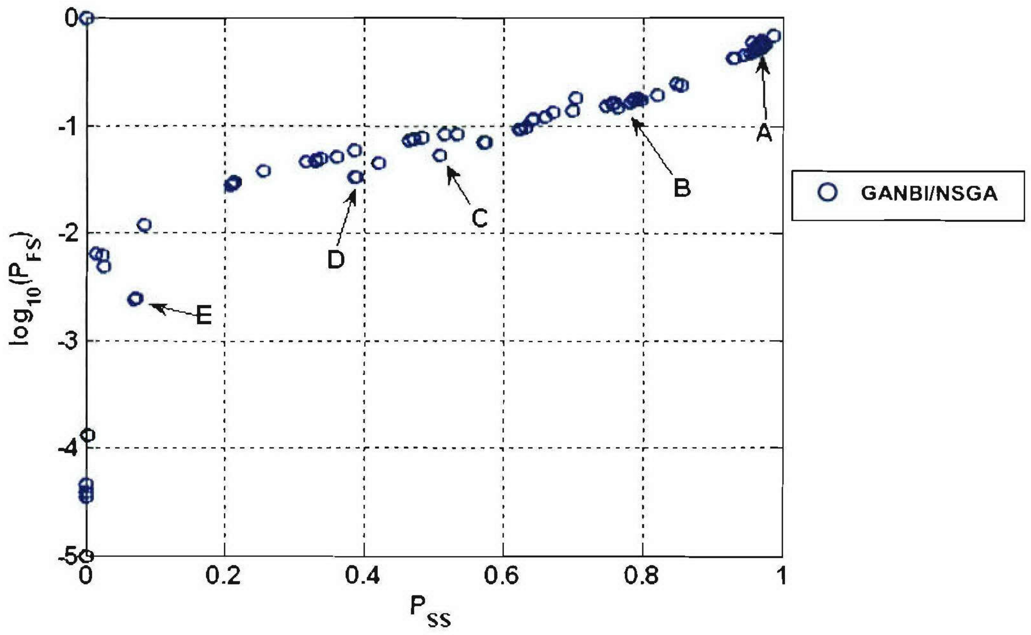

Figure 11. Approximate Pareto Sets Obtained Using GANBI/NSGA for Problem DNS with Sample Results Marked A, B, C, D, E

Table 2. Parameter and Objective Values for Selected Points in Figure 11

Point	r_d (meters)	m	γ	P_{SS}	P_{FS}
A	313	985	2	0.97	0.52
B	309	585	4	0.76	0.15
C	299	396	1	0.51	0.054
D	281	348	3	0.39	0.033
E	314	131	3	0.07	0.002

CONCLUSIONS

A new algorithm for performing the Pareto optimization of engineering design problems has been presented. The algorithm functions as a preprocessor for conventional multiobjective genetic algorithms and uses a small sampling of the design space to provide reasonable estimates of the Pareto set of designs. By functioning with genetic algorithms, each iteration leads to a family of designs that approximates the Pareto set (as opposed to a single design at each iteration). As the iterations progress, the resulting family of designs begins to converge to the true Pareto set. The algorithm is based on the concept of normal boundary intersection and provides a robust estimate of the Pareto set with a predefined spread of solution points at each iteration. The new algorithm has been named the genetic-algorithm-based normal boundary intersection (GANBI) method.

The GANBI method was shown to provide better performance than traditional multiobjective optimization solvers on an academic problem. The performance of the GANBI method when used as a preprocessor for a standard solver (VEGA) and a newer state-of-the-art solver (NSGA) was compared with the performance of VEGA and NSGA used alone. Application of the GANBI method as a preprocessor improved the performance of both of these solution techniques. Furthermore, the GANBI solver required a very small sampling of the overall design space, which makes it an effective tool for providing design guidance in multiobjective situations. To demonstrate its effectiveness in practical multiobjective engineering design, the

GANBI method was applied to a design problem for distributed sensor networks—an example of interest. The choice of sensor range, numbers of sensors, and number of multiple sensor detections required by a system were provided as design parameters. GANBI effectively mapped the approximate Pareto set of these design parameters for the bi-objective problem of maximizing search effectiveness while minimizing false searches. More complex problems of many different types can be tackled with the GANBI method. This report provides only an overview of the method and an illustration of how it can be used.

REFERENCES

1. P. Y. Papalambros and D. J. Wilde, *Principles of Optimal Design: Modeling and Computation*, Cambridge University Press, Cambridge, UK, 2000.

2. G. P. Liu, J. B. Yang, and J. F. Whidborne, *Multiobjective Optimisation and Control*, Research Studies Press Ltd., Baldock, Hertfordshire, England, 2003.

3. K. Miettinen, *Nonlinear Multiobjective Optimization*, Kluwer, Norwell, MA, 1998.

4. K. Deb, *Multiobjective Optimization Using Evolutionary Algorithms*, Wiley, Chichester, UK, 2001.

5. C. A. Coello Coello, "An Updated Survey of Evolutionary Multiobjective Optimization Techniques: State of the Art and Future Trends," *Proceedings of the Congress on Evolutionary Computation*, IEEE Press, vol. 1, 1999, pp. 3–13.

6. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Boston, 1989.

7. D. A. Van Veldhuizen, "Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations," Ph.D. Dissertation, Air Force Institute of Technology, 1999.

8. D. A. Van Veldhuizen and G. B. Lamont, "Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art," *Evolutionary Computation*, vol. 8, no. 2, 2000, pp. 125–147.

9. E. Zitzler and L. Thiele, "Multiobjective Optimization Using Evolutionary Algorithms – A Comparative Case Study," *Proceedings of Parallel Problem Solving from Nature – PPSN V*, Amsterdam, The Netherlands, September 1998, pp. 292–301.

10. I. Das and J. E. Dennis, "A Closer Look at Drawbacks of Minimizing Weighted Sums of Objectives for Pareto Set Generation in Multicriteria Optimization Problems," *Structural Optimization*, vol. 14, 1997, pp. 63–69.

11. I. Das and J. E. Dennis, "Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems," *SIAM Journal of Optimization*, vol. 8, no. 3, 1998, pp. 631–657.

12. V. Pareto, *Manual of Political Economy* (English translation by A. S. Schwier), MacMillam Press, New York, 1971.

13. J. D. Schaffer, "Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms," Ph.D. Dissertation, Vanderbilt University, 1984.

14. J. Horn, N. Nafpliotis, and D. E. Goldberg, "A Niched Pareto Genetic Algorithm for Multiobjective Optimization," *Proceedings of the First IEEE Conference on Evolutionary Computation (ICEC '94)*, Piscataway, NJ, vol. 1, 1994, pp. 82–87.

15. N. Srinivas and K. Deb, "Multiobjective Function Optimization Using Nondominated Sorting Genetic Algorithms," *Evolutionary Computation*, vol. 2, no. 3, 1995, pp. 221–248.

16. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, 2002, pp. 182–197.

17. I. Das, "Nonlinear Multicriteria Optimization and Robust Optimality," Ph.D. Dissertation, Rice University, 1997.

18. T. A. Wettergren, "Performance of Search Via Track-Before-Detect for Distributed Sensor Networks," manuscript under journal review, 2006.

19. C. M. Traweek and T. A. Wettergren, "Efficient Sensor Characteristic Selection for Cost-Effective Distributed Sensor Networks," *IEEE Journal of Oceanic Engineering*, vol. 31, no. 2, 2006.

INITIAL DISTRIBUTION LIST

Addressee	No. of Copies
Office of Naval Research (ONR 321 – M. Traweek, M. Wardlaw)	2
Center for Naval Analyses	1
Defense Technical Information Center	2