5-2018

A Multi-Fidelity Approach to Address Multi-Objective Constrained Mixed-Discrete Nonlinear Programming Problems With Application to Greener Aircraft Design

Samarth Jain
Purdue University

A MULTI-FIDELITY APPROACH TO ADDRESS MULTI-OBJECTIVE CONSTRAINED MIXED-DISCRETE NONLINEAR PROGRAMMING PROBLEMS WITH APPLICATION TO GREENER AIRCRAFT DESIGN

A Thesis

Submitted to the Faculty

of

Purdue University

by

Samarth Jain

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Aeronautics and Astronautics

May 2018

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF THESIS APPROVAL

Dr. William A. Crossley, Chair

School of Aeronautics and Astronautics

Dr. Dengfeng Sun

School of Aeronautics and Astronautics

Dr. Jitesh Panchal

School of Mechanical Engineering

Approved by:

Dr. Weinong Chen

Head of the School Graduate Program

Dedicated to Shanu Jain and Kapil Jain.

ACKNOWLEDGMENTS

I would like to thank the members of my advisory committee, Dr. William A. Crossley, Dr. Dengfeng Sun, and Dr. Jitesh Panchal. I would especially like to thank my committee chair and advisor, Dr. Crossley, for guiding and supporting me throughout my graduate life at Purdue, for funding my graduate program, and for giving me the opportunity to work on another project along with my thesis research which helped my learning curve grow exponentially. I would like to thank Dr. Sun and Dr. Panchal for their guidance and valuable feedback throughout my work.

I would like to thank Dr. Satadru Roy for taking out his valuable time to continuously guide me through my research work, and also for reviewing my thesis document. I would also like to thank Nithin Kolencherry, Kshitij Mall, and Soumya Roy, for reviewing my thesis document and providing their valuable feedback.

A special thanks goes to my family, my parents and my brother, for their unconditional love and support throughout my graduate life. Also, a special mention goes to Vidushi for always motivating me and supporting me. Special appreciation goes to all my friends at Purdue for making this journey memorable.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Jain, Samarth. M.S.A.A., Purdue University, May 2018. A Multi-Fidelity Approach to Address Multi-Objective Constrained Mixed-Discrete Nonlinear Programming Problems With Application to Greener Aircraft Design. Major Professor: William A. Crossley.

Engineering problems often involve solving constrained multi-objective Mixed-Discrete Nonlinear Programming (MDNLP) problems. These problems are inherently difficult to solve given the presence of multiple competing objectives, nonlinear objective and constraint functions, mixed-discrete type design variables, and expensive analysis tools. This work presents a multi-fidelity approach that addresses all these features together and exhibits its efficacy to solve constrained multi-objective MDNLP problems within a reasonable computational budget. The work addresses the high computational cost drawback associated with a previously developed "hybrid multi-objective optimization approach" that combines a Genetic Algorithm (GA) with the gradient-based Sequential Quadratic Programming (SQP) algorithm. The multi-fidelity hybrid algorithm in this work employs surrogate models to provide low-fidelity approximations of the objective and constraint functions that are fast to evaluate. The gradient-based SQP algorithm uses these surrogate models in a goal attainment formulation. The combination of the GA with SQP then finds a diverse set of designs representing the best possible trade-off solutions for the multi-objective problem. For this thesis, the author initially pursues both Kriging and Radial Basis Function (RBF) surrogate modeling techniques, with their respective application to test problems (three-bar and ten-bar truss constrained, multi-objective, MDNLP problems) determining their feasibility of implementation in the multi-fidelity approach. The test problem results indicate that using RBF technique makes use of the hybrid approach more feasible as compared to using the Kriging technique. The re-

sults show a reduction of at least 98% in the "high-fidelity" function evaluations with respect to the previously-developed hybrid approach, along with a reduction of at least 89% in the computational runtime. Subsequently, the multi-fidelity approach using RBF surrogate models is employed to solve a complex aerospace engineering problem used in previous studies – a 'greener' aircraft design problem – posed as a constrained multi-objective MDNLP problem. The resulting non-dominated design solutions are comparable to those obtained using the previously-developed hybrid approach. The result indicates a compromise that exists between the number of "high-fidelity" evaluations performed and the ability of the multi-fidelity hybrid algorithm to find as diverse non-dominated designs as possible (indicating the spread of the Pareto frontier). This work also suggests a preliminary approach to choose the population size for the multi-objective multi-fidelity hybrid algorithm, so that the algorithm finds a satisfactory spread for the Pareto frontier at a reasonable computational cost.

1. INTRODUCTION

Engineering simulations and analyses can be categorized into different levels of "fidelity" based upon the accuracy of representation of the physics of the problem through computer programs using mathematical models. "High-fidelity" analyses involve fewer assumptions about the physics governing the problem. "Higher-fidelity" usually leads to more accurate calculations but requires longer setup and execution times, ultimately increasing the computational cost of the analyses. On the other hand, "lower-fidelity" analyses tend to have shorter run times (computationally cheaper to evaluate), but with a lesser detailed depiction of the physics and more assumptions in the analysis. In general, the "high-fidelity" analyses tend to be expensive for optimization due to their high computational cost.

Constrained multi-objective mixed-discrete problems are inherently difficult to solve given the presence of multiple competing objectives, nonlinear objective and constraint functions, mixed-discrete and continuous type design variables, and computationally expensive analysis tools. There is published work that tries to address some of these aspects simultaneously [1–10], but limited works exist that try to address all of these issues concurrently [11]. Although the hybridization of Genetic Algorithm (GA) with gradient-based search seems promising for constrained multi-objective mixed-discrete problems, using expensive "high-fidelity" analyses with this approach limits its applicability to a wider domain of engineering optimization problems. The work here proposes a multi-fidelity approach to solve constrained multi-objective Mixed-Discrete Nonlinear Programming (MDNLP) problems and to find these solutions within a reasonable computational budget. The effort here combines surrogate-based approximation techniques with a previously developed hybrid approach that couples the design space exploration capability of the GA with the computational efficiency of a gradient-based Sequential Quadratic Programming (SQP)

algorithm. The motivation for this research directly comes from the need to reduce the computational cost incurred in solving multi-objective engineering optimization problems.

The proposed multi-fidelity approach seeks to find optimum design solutions by utilizing the hybrid approach as the base for design optimization. Design optimization involves using numerical methods to solve design problems. This includes performing design iterations and analyses using optimization algorithms, which work to find the best combination of design variables that lead to optimized designs while satisfying problem constraints.

Many engineering design problems are multi-objective in nature. Multi-objective problems require simultaneous optimization of two or more competing objectives. There exists no single meaningful solution to such problems, rather there exists a range of best possible solutions amidst all objectives. This set of best possible solutions is called the Pareto-optimal set. Several previous efforts have shown that the population-based Evolutionary Algorithms (EAs) are capable of generating a good representation of the Pareto-optimal set of designs [12, 13]. This capability of evolutionary algorithms to handle multiple solutions simultaneously makes them suitable for solving multi-objective problems.

GA is a well-known class of population-based EA, which shows capability to explore the entire design space and locate the near-global optimal design solution. In addition, GA can easily handle both continuous and discrete design variables, making it a plausible choice for solving MDNLP problems. GA provides a near-global optimum solution for a problem, because of its probabilistic, not calculus-based, search. Different GA runs can find different optimum solutions, but usually these solutions are similar. GA cannot directly enforce constraints, because it relies on a penalty approach to account for violated constraints.

On the other hand, SQP is a well-known gradient-based search algorithm that converges to a local optima while directly handling problem constraints. SQP is computationally efficient, because it relies on gradient information to find a local

minima (or maxima), which includes satisfaction of constraints. However, unlike GA, SQP cannot handle discrete variables.

To overcome the limitations of both, the population-based search algorithm - GA, and the gradient-based search algorithm – SQP, a hybrid approach combines both these algorithms to fully address constrained multi-objective problems that comprise both continuous and discrete variables. The hybrid approach uses a population of designs from GA, and the "fitness evaluation" of each design involves the use of SQP to solve a gradient-based version of the problem. This requires the hybrid approach to conduct many "high-fidelity" function evaluations, via the gradient-based local search, to find diverse trade-off points representing the different problem solutions. To overcome this limitation, the work here employs a surrogate modeling approach to provide "low-fidelity" approximations of the objective functions and constraint functions in the local search step, reducing the number of "high-fidelity" function evaluations required for solving MDNLP problems using a hybrid approach.

Surrogate models are analytic models that approximate objective / constraint function values for different combinations of design variables, based on a limited set of computationally expensive ("high-fidelity") analyses. These models have a characteristic advantage of reducing the computational cost associated with complex simulations by predicting their values at different points in the design space. However, as these surrogate models are approximations, the predictions will include modeling error.

The hybrid optimization approach (GA in conjunction with SQP) and the surrogate modeling techniques have inherent advantages and disadvantages associated with their applicability to optimization problems. Combining surrogate modeling techniques with the hybrid approach leads to a novel multi-fidelity algorithm to solve constrained multi-objective MDNLP problems within a limited computational budget. To demonstrate the efficacy of this proposed multi-fidelity approach to solve complex aerospace engineering problems with reduced computational cost, the work here applies this multi-fidelity algorithm to re-solve the 'greener' aircraft design prob-

lem presented in Refs. [1–3, 14]. The aircraft design problem serves as a plausible example of a constrained multi-objective MDNLP engineering problem. The same problem is re-solved to establish a quantitative and qualitative basis for comparison with the standalone hybrid approach.

The aircraft design problem intends to illustrate the consequences of including 'greener' technologies (i.e., ones that reduce environmental impact) in a short-to-medium range commercial aircraft. These 'greener' technologies include composite structures, natural laminar flow, and hybrid laminar flow. The goal of solving this 'greener' aircraft problem is to demonstrate the ability of the proposed multi-fidelity hybrid algorithm to consider the discrete technologies in addition to continuous variables, so that the resulting designs are the best possible range of aircraft trade-offs. The technologies used in aircraft along the Pareto frontier are those that hold promise for further investigation in the near future. With the goal to search for a 'greener' aircraft while studying the interactions between the various environmental, economic, and performance metrics, the competing objectives for the problem include the total fuel carried (for every pound of fuel consumed, the engines produce about 3.2 pounds of CO_2, making the total fuel carried an index of aircraft CO_2 emissions), emissions of nitrogen oxides (NO_X), and the total operating cost.

2. OPTIMIZATION TECHNIQUES

Design optimization is the process of finding an optimal combination of design variables that minimize (or maximize) the objective function(s) while satisfying all the constraints in the design space. The definitions of the terms associated with optimization are as follows:

Objective function: The function to be minimized (or maximized) in the optimization problem, e.g., cost, weight, etc. An objective function may be uni-modal with just one optimal solution, or multi-modal with multiple locally-optimal solutions and a global optimal solution.

Constraint function: The restrictions/bounds that must be satisfied to produce a feasible design, e.g., allowable stress, maximum displacement, etc. Design constraints are also functions of the design variables. A problem could have equality or inequality constraints, or maybe both.

Design variables: The quantities that describe a design. A change in the design variables alters the design, changing its objective and constraint function values. Design variables can be continuous, discrete, or mixed-discrete continuous in nature.

Feasible design: A design that satisfies all constraints.

The following expression shows the mathematical formulation for a general optimization problem.

Minimize:

$$\begin{aligned} & f(\mathbf{x}) \quad (Single-objective\ formulation) \\ & \mathbf{f}(\mathbf{x}), \quad where\ \mathbf{f} = \{f_1, f_2, ..., f_n\} \quad (Multi-objective\ formulation) \end{aligned} \tag{2.1}$$

Subject to:

$$\begin{aligned} g_j(\mathbf{x}) \leq 0,\ j = 1, ..., m \qquad & (\textit{inequality constraints}) \\ h_k(\mathbf{x}) = 0,\ k = 1, ..., l \qquad & (\textit{equality constraints}) \\ x_i^L \leq x_i \leq x_i^U,\ i = 1, ..., p \qquad & (\textit{bound constraints}) \end{aligned} \tag{2.2}$$

where $\mathbf{x} = [\ x_1,\ x_2,\ ...,\ x_p]^T$.

2.1 Hybrid Optimization Approach

This work employs the hybrid optimization approach presented in Ref. [1–3] to solve constrained multi-objective MDNLP problems. The multi-objective hybrid algorithm combines the capability of an evolutionary algorithm to explore the whole design space, while handling discrete variables, with the local optima searching and constraint handling capabilities of a gradient-based search. This hybrid technique exploits modified two-branch tournament GA [1–3] as the evolutionary algorithm, and SQP with goal-attainment technique as the gradient-based search method. The two-branch tournament GA [15] compares designs with respect to both of the problem objectives one by one in a two-step process. The two-branch tournament GA is explained in detail in Section 2.3.2.1, while the modified two-branch tournament GA appears in Section 3.1.2.1.

In this hybrid approach, GA acts like a guide for a multi-start approach through how it combines the discrete and continuous design variables. The local search can be considered as "learning" that takes place in an individual design during every GA generation [1–3]. The two-branch tournament GA includes both the continuous and discrete variables in the representation of each design. The continuous variable values in the GA individual are used as initial points for local search using a goal-attainment problem with SQP algorithm, which essentially converts the multi-objective problem into a single-objective optimization problem. The goal-attainment algorithm seeks objective values as close as possible to a set of predefined objective goal values, without violating any of the problem constraints. The objective values of the solutions

obtained from the local search problem are returned to the GA-level for use in the modified two-branch tournament selection [1–3]. A single-objective version of this hybrid approach appears in [16], which combines a binary coded GA with SQP algorithm as the hybrid optimizer to solve constrained single-objective MDNLP problems.

The following Sections 2.2 and 2.3 provide a review of gradient-based optimization (with a focus on SQP algorithm and goal attainment formulation), evolutionary algorithms (with a focus on single-objective GA), and multi-objective GA formulation (two branch tournament GA), all of which form an integral part of the proposed multi-fidelity hybrid algorithm. Section 2.4 discusses the limitations of the hybrid approach, which acts as a motivation to combine the hybrid approach with approximation/surrogate models.

2.2 Gradient-Based Optimization

Gradient-based optimization strategies usually follow an iterative approach. An initial set of design variables, $\mathbf{x}^0$, are utilized to find the optimal set of design variables by updating the design variables in every optimization iteration. The following mathematical expression depicts the iterative approach [17]:

$$\mathbf{x}^q = \mathbf{x}^{q-1} + \alpha^* \mathbf{S}^q \tag{2.3}$$

where q denotes the iteration number, $\mathbf{S}^q$ denotes the search direction in the design space, and α^* denotes the step length. The chosen search direction, $\mathbf{S}^q$, should lead to a feasible direction without usually violating any constraints, ensuring that the next design point gets closer to the optimal design. For instance, the steepest descent method uses the negative of the objective function gradient as its search direction, $\mathbf{S}^q$. With $\mathbf{S}^q$ known, the problem is one-dimensional, requiring an estimation of the step length, α^*. The step length signifies the distance moved along the search direction to find the optimal design without violating any of the constraints. Hence, a nonlinear gradient-based optimization algorithm can be split into two components: 1) determining the search direction $\mathbf{S}$, and 2) determining the step length, α^*.

2.2.1 Determining the Search Direction

Many methods exist to estimate the search direction for determining the optimal design [17]. These methods can be classified as zero-order, first-order, and second-order methods, based on the order of objective function derivative required for the method.

A zero-order method finds the optimal design solution, $\mathbf{x}^*$, by evaluating the objective function, $f(\mathbf{x})$, at a large number of random initial design points $\mathbf{x}$. A well-known zero-order method is Powell's method [17], which uses successive conjugate steps to approximate the Hessian matrix. The Hessian matrix contains the second-order partial derivatives of the objective function with respect to the design variables.

The first-order methods use gradient information (first order partial derivatives) to find the search direction. The Steepest Descent method uses negative of the gradient of the objective function as the search direction for an unconstrained search. More sophisticated first-order methods are derived from this method. The conjugate direction method uses search directions that are conjugate to each other. The convergence rate of this method is significantly greater (faster) than to the Steepest Descent method. Other popular unconstrained first-order methods include variable metric methods like the Broyden-Fanno-Goldfarb-Shanno (BFGS) and Davidon-Fletcher-Powell (DFP) method.

Newton's method is a second-order method that uses both first- and second-derivative information of the objective function. In this method, the search direction results from finding $\mathbf{S}$ so that $\nabla f(\mathbf{x} + \mathbf{S}) = \mathbf{0}$. If the objective function is a quadratic function, then there is no need to find a step length. For a general function, $f(\mathbf{x})$, Newton's method does find a search direction, $\mathbf{S}$. Because of the second-order information, Newton's method is more efficient than the zero-order or first-order methods. For computational consideration, the Hessian matrix is not updated every generation using an assumption that this matrix does not show any drastic changes between a

few successive iterations, reducing the computational burden associated with Newton's method for every iteration.

2.2.2 Determining the Step Length

The step length is determined by solving a one-dimensional optimization problem - minimizing the function value along the search direction using step length, α, as the design variable. The most popular techniques for solving this problem are polynomial approximation and Golden search technique [17]. The polynomial approximation technique models the objective function using a polynomial curve fit. It finds the value of α^* for which the first derivative of the polynomial model of the objective function is zero. The Golden section search technique works by dividing the design space into smaller portions until the optimal solution is found. This method even works for functions that do not possess continuous derivatives.

2.2.3 Constraint Handling

Constrained optimization problems can be solved using any unconstrained optimization algorithm by the addition of a penalty function to the original objective function. The pseudo-objective function, $\phi(\mathbf{x})$, is expressed as follows:

$$\phi(\mathbf{x}) = f(\mathbf{x}) + r_p P(\mathbf{x}) \tag{2.4}$$

where $P(\mathbf{x})$ is the penalty function, and r_p is the penalty multiplier. The value of r_p is kept constant for a complete unconstrained minimization [17]. There are three main types of penalty functions: exterior penalty function, interior penalty function, and extended interior penalty function.

Exterior penalty functions are the easiest to implement and penalize the objective function only when a constraint is violated. These functions are applicable to both inequality and equality constraints.

Interior penalty functions penalize the objective functions that approach the feasible design space boundary internally. This ensures that there is absolutely no constraint violation. These functions can only be used to enforce inequality constraints and require a feasible starting point.

Extended interior penalty functions combine the advantages of exterior and interior penalty functions. The penalty function behaves as an interior penalty function when the constraint value is less than a small negative value ϵ. For constraint values greater than ϵ, the function becomes a linear extended penalty function.

2.2.4 Optimality Criteria – The Karush-Kuhn-Tucker Conditions

The Karush-Kuhn-Tucker conditions for constrained problems define the necessary conditions for a design to be optimal. The Lagrangian function accounts for the objective function and the constraint functions in a single equation. The Lagrangian function is expressed as:

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \sum_{j=1}^{m} \lambda_j g_j(\mathbf{x}) + \sum_{k=1}^{l} \lambda_{m+k} h_k(\mathbf{x}) \tag{2.5}$$

For an optimal design $\mathbf{x}^*$, the following Karush-Kuhn-Tucker conditions must be satisfied:

1. $\mathbf{x}^*$ is feasible

2. $\lambda_j g_j(\mathbf{x}^*) = 0 \; \forall \; j = 1,m \; \& \; \lambda_j \geq 0$

3. $\nabla f(\mathbf{x}^*) + \sum_{j=1}^{m} \lambda_j \nabla g_j(\mathbf{x}^*) + \sum_{k=1}^{l} \lambda_{m+k} \nabla h_k(\mathbf{x}^*) = 0$

$\lambda_j \geq 0$

λ_{m+k} unrestricted in sign

Condition 1 states that the optimum design $\mathbf{x}^*$ must satisfy all the problem constraints. Condition 2 states that if constraint $g_j(\mathbf{x})$ is not active (i.e., $g_j(\mathbf{x}^*) < 0$), then the corresponding Lagrange multiplier, λ_j, must be zero. Condition 3 states

the gradient of the Lagrangian function should be zero at the optimal point. This implies that at the optimal point, a linear combination of the gradient of the objective function and the gradients of the constraints must equal zero.

For unconstrained problems, the optimal solution is found when the gradient of the objective function is equal to zero. The second-order derivative of the objective function with respect to the design variables (known as the Hessian matrix) describes the curvature of the objective function, stating whether the optimal solution is maxima or minima. For the minimum of a function, the Hessian matrix will always be positive definite, implying that all its eigenvalues will be greater than zero. However, this does not guarantee that the optimal solution will be a global minimum. Hence, one of the drawbacks of gradient-based optimization is that it does not guarantee a global optimal solution; for problems that have more than one local optimum, the solution obtained by the gradient-based search is dependent upon the chosen starting design.

2.2.5 Sequential Quadratic Programming

This work exploits SQP for the local search portion of the multi-fidelity hybrid algorithm. SQP is a well-known computationally efficient gradient-based technique that outperforms other gradient-based techniques for solving constrained optimization problems. The comparison of SQP with other gradient-based algorithms appears in Ref. [18]. However, as with every gradient-based technique, SQP finds a local minima depending on the starting point.

The basic algorithm for SQP can essentially be divided into two parts: First, the algorithm finds the search direction, $\mathbf{S}$, by approximating the Lagrangian function for the constrained problem as a quadratic function and then minimizes this approximating function with linearized constraints using quadratic programming. Second, the search direction so obtained is used to minimize the augmented Lagrangian to find the step length, α^*. The BFGS approach updates the approximation to the quadratic

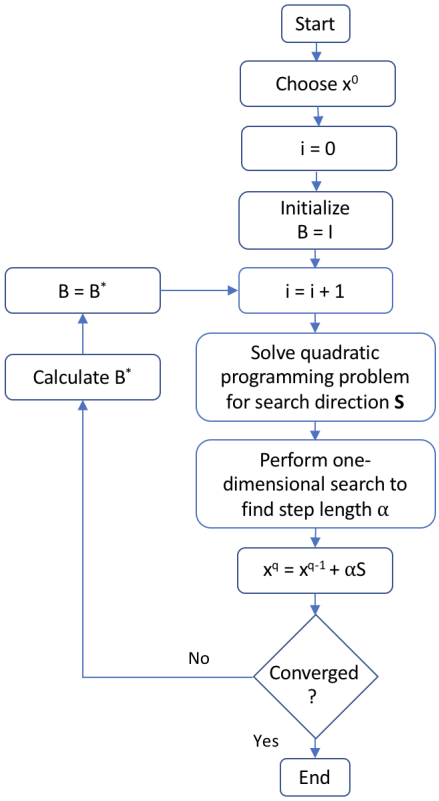

Figure 2.1. Flowchart for Sequential Quadratic Programming.

Lagrangian function. These two basic steps are discussed in detail in the following paragraphs.

SQP technique finds the search direction by solving a subproblem with a quadratic approximation to the augmented objective function, and a linear approximation to the constraints [17]. The subproblem is expressed as follows:

Minimize:

$$Q(\mathbf{S}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^T \mathbf{S} + \frac{1}{2}\mathbf{S}^T \mathbf{B}\mathbf{S} \tag{2.6}$$

Subject to:

$$\begin{aligned} \nabla g_j(\mathbf{x})^T \mathbf{S} + \delta_j g_j(\mathbf{x}) \leq 0, \;\; j = 1, m \\ \nabla h_k(\mathbf{x})^T \mathbf{S} + \bar{\delta} h_k(\mathbf{x}) \leq 0, \;\; k = 1, l \end{aligned} \tag{2.7}$$

where $\mathbf{S}$ is the search direction, $\mathbf{B}$ is a positive definite update matrix which is initially posed as an identity matrix. Matrix $\mathbf{B}$ is updated in subsequent iterations to

approximate eventually the Hessian of the Lagrangian function. The scalar parameters, δ and $\bar{\delta}$, are usually problem dependent and are added to ensure that there is a feasible space when using the linearized constraints. These parameters are defined as:

$$\begin{gathered} \delta_j = 1, \ g_j(\mathbf{x}) < 0 \\ \delta_j = \bar{\delta}, \ g_j(\mathbf{x}) \geq 0 \\ 0 \leq \bar{\delta} \leq 1 \end{gathered} \tag{2.8}$$

With the search direction known, the SQP algorithm now calculates the step length, α, using a one-dimensional search problem. The search problem here employs an augmented Lagrangian function, ϕ, with an exterior penalty function to convert to an unconstrained problem. The problem is expressed as follows:

$$\phi = f(\mathbf{x}) + \sum_{j=1}^{m} u_j \{max[0, g_j(\mathbf{x})]\} + \sum_{k=1}^{l} u_{m+k} |h_k(\mathbf{x})| \tag{2.9}$$

where, $\mathbf{x} = \mathbf{x}^{q-1} + \alpha \mathbf{S}$,

$u_j = |\lambda_j|$, j=1,m+l for first iteration,

$u_j = \max[|\lambda_j|, \frac{1}{2}(u_j + |\lambda_j|)]$ for subsequent iterations, and,

$u'_j = u_j$ from the previous iteration. This one-dimensional problem is well-conditioned and usually $\alpha = 1.0$ is a very good initial estimate for α^* [17].

Now once the search direction, $\mathbf{S}$, and step length, α^*, are known for updating the design, the SQP technique updates the matrix $\mathbf{B}$ for use in the subsequent iteration. Ref. [19] recommends the Broyden-Fanno-Goldfarb-Shanno (BFGS) update formula for this task. A flowchart for the SQP technique appears in Fig. 2.1.

2.2.6 Gradient-Based Methods for Multi-Objective Optimization

Multi-objective optimization requires simultaneous optimization of two or more competing objectives. These problems do not possess a single optimal solution. Rather, there exists a range of possible optimal solutions amongst all the objectives called the Pareto-optimal set, named after Vilfredo Pareto [20]. The Pareto set

comprises the Pareto-optimal designs. A trade-off curve representing these designs forms the Pareto frontier. Mathematically, all of these Pareto-optimal designs are non-dominated. A non-dominated solution is a design such that there is no improvement possible in any of the objective function values without degrading some of the other objective values. A solution x_i dominates another solution x_j only if the solution x_i performs better than or equal to x_j in all objectives, plus is strictly better than x_j in at least one objective. This relationship between the dominating design, x_i, and the dominated design, x_j, can be expressed as:

$$\begin{aligned} f_l(x_i) &\leq f_l(x_j),\ l = 1, ..., L \\ f_l(x_i) &< f_l(x_j),\ l \in [1, L] \end{aligned} \tag{2.10}$$

where L is the number of objectives. A Pareto-optimal solution cannot be dominated by any other solution in the design space.

For multi-objective optimization using gradient-based methods, the multiple objectives in the problem need to be "scalarized". This process can be undertaken using any one of the three approaches – 1) weighted sum approach, 2) ϵ-constraint approach, and 3) goal-attainment formulation. This work implements the goal-attainment approach to solve the multi-objective local search problem embedded in the multi-fidelity hybrid approach. The next few paragraphs explain the weighted sum approach and the ϵ-constraint approach briefly for context. A detailed explanation of the goal-attainment approach follows.

The weighted sum approach converts multiple objectives into a single objective by assigning weights to each of the objectives and then adding together the products of each weight coefficient and its corresponding objective function. The result is a single objective function. Any of the previously discussed gradient methods can solve the converted single objective optimization problem; if the original problem had constraints, the gradient-based algorithm must also handle these original constraints. For a specific set of weights, the optimal problem solution will lead to a single point on the Pareto frontier. Hence, the problem needs to be solved with different combinations of weights to find multiple points on the Pareto frontier. This process can be

problematic, because it is very difficult to compare different functions without proper scaling and identify their relative weights. Also, the weighted sum approach cannot find Pareto optimal designs in regions of the Pareto frontier where the trade-off between objectives is non-convex. Additional details of the weighted sum approach appear in sources like Ref. [21].

The ϵ-constraint approach addresses multi-objective optimization by converting one of the multiple objectives into a single primary objective and incorporating the other objectives as inequality constraints that limits the maximum value these other objectives can have (assuming that all objectives are minimized). This approach can handle both convex and non-convex Pareto frontiers. The limiting constraint values ϵ_l are user-defined, and the epsilon-constraint problem must be solved multiple times, each with a different set of ϵ_l values, to find different solutions on the Pareto frontier. Also, a prior knowledge of the design space is often important for this approach, because the chosen ϵ_l values need to be within the range of possible values of their corresponding objective function values, for the approach to find feasible designs.

2.2.6.1 Goal Attainment Formulation

The goal attainment formulation solves a multi-objective problem by working to attain specific user-defined goal values for the multiple objectives, f_l^G. This technique minimizes a goal attainment factor, γ, instead of a weighted or primary objective function. In this approach, the multiple objective functions are converted into a set of inequality constraints using the goal values, f_l^G. Any other inequality or equality constraints in the problem are included alongside these objective-goal constraints.

Solving the goal attainment problem brings the optimal design point as close as possible to the desired goal point by minimizing the attainment factor, γ, along the direction of the weight vector, $\mathbf{w}$, while satisfying constraints. The weight vector signifies the relative importance of each objective in attaining the goal point. The mathematical formulation of this technique is as follows:

Minimize:

$$\gamma \tag{2.11}$$

Subject to:

$$\begin{aligned} f_l(\mathbf{x}) - w_l\gamma \leq f_l^G, \quad & l = 1, ..., L \\ g_j(\mathbf{x}) \leq 0, \quad & j = 1, ..., J \\ h_k(\mathbf{x}) = 0, \quad & k = 1, ..., K \end{aligned} \tag{2.12}$$

The goal attainment technique requires two types of user-defined inputs - the goal points and the weight vector. Similar to the weighted sum and ϵ-constraint approach, there exists a single point on the Pareto frontier for every goal attainment problem solution for each combination of $\mathbf{f}^G$ and $\mathbf{w}$ values.

2.2.6.2 Comments about Gradient-Based Multi-Objective Approaches

Gradient-based multi-objective approaches are fairly effective in solving multi-objective optimization problems. The gradient-based methods can be fast to solve, they can meet KT conditions and find at least "weakly" Pareto optimal solutions, if not "strongly" Pareto optimal solutions. However, they depend on a number of user-defined input values and require multiple solutions to find multiple points on the Pareto frontier.

For a continuous problem that can be solved via a gradient-based method, finding lots of Pareto-optimal solutions via multiple solutions is still most always faster than the population-based EA / GA approaches. However, if the problem requires use of a GA, like the constrained mixed-discrete nonlinear problem, then the hybrid approach becomes appealing. The implementation of a population-based EA / GA approach can often remove the need for a user-defined set of weights.

2.3 Population-Based Optimization Algorithms

Evolutionary algorithms (EA) are global optimization algorithms that depend on a population-based search to find a globally optimal solution. EAs do not require calculation of any function derivative, making them zero-order methods that depend only on function values calculated at different design points. EAs can solve problems with discontinuous functions and still near-globally optimum design solution – which the gradient based optimization techniques cannot. Further, when using a population-based search for problems, the population can provide a way to find multiple non-dominated solutions for multi-objective problems in a single run of the search algorithm.

The Genetic Algorithm (GA) is one of the most well-known class of population-based EA that finds its application in engineering design, game theory, machine learning, numerical optimization, etc [22]. The main difference between EA and GA lies in the fitness assignment techniques, elitism and the methods to obtain a diversified solution. Specifically, EA relies upon selection and almost entirely on mutation for its search whereas GA relies upon selection and mostly crossover for its search. This work employs a variant of GA as the global search component of a hybrid method to solve constrained multi-objective MDNLP problems. The following sections discuss single-objective GA and multi-objective GA variants in detail.

2.3.1 Genetic Algorithm

GA is a computational model of the evolution displayed by natural populations. Holland [23] and his students developed this algorithm in 1960s and 1970s as a computational representation of the natural selection process. Since then, GA has been utilized for solving optimization problems. GA is inspired by Darwin's "Theory of Natural Selection", which advocates the concept of survival of the fittest. For GA, survival of the fittest acts as an analog to the selection of a better design in an optimization algorithm [22]. This analogy also includes representing the designs as

a population, performing selection (survival of the fittest), mutation, and crossover (reproduction) of designs in the current population to create new designs (offspring) that form the next population. GA differs from the gradient-based optimization approaches as it follows a probabilistic search instead of a calculus-based search, providing a near-global optimum solution, $\mathbf{x}^*$. There is no mathematical proof of convergence of the algorithm to a global solution; hence, the "near-global" modifier. The Karush-Kuhn-Tucker optimality cannot be established because the algorithm has no gradient information available. GA requires no initial starting point – evaluating all designs in the population at the same time. Instead of using the actual design variables, GA uses coding of the design variables – usually binary (0s and 1s) and in some cases, real numbers. Each encoded design variable string represents a gene; these genes are linked together to form a chromosome that represents an individual in the population. This coding equips GA with the ability to handle continuous, integer, and discrete design variables. Because there are some random numbers used in the operators for the global search, different GA runs can find different $\mathbf{x}^*$, but usually these $\mathbf{x}^*$ are similar. GA is computationally expensive, because it evaluates every individual in each generation. The computational expense associated with GA limits its applicability to problems with "high-fidelity" function evaluations that require even moderate amounts of computational time.

2.3.1.1 Fitness Function and Constraint Handling

GA is naturally suited to solve unconstrained optimization problems, because it uses a single fitness function value to drive the search via the selection operator. For constrained problems, a penalty addition approach is the most widely used technique to take care of any problem constraints. For implementing this technique, the fitness function for GA must reflect all objectives and constraints. The penalty constraint handling approach adds penalties to the fitness function for violation of any inequality or equality constraints. For constrained problems, the fitness function, ϕ, is given by

the addition of $g_m(\mathbf{x})\leq 0$ (equality constraints) and $h_l(\mathbf{x})=0$ (inequality constraints) to $f(\mathbf{x})$ via an exterior quadratic penalty function. The expression for the fitness function ϕ appears below:

$$\phi(\mathbf{x}) = f(\mathbf{x}) + r_p \sum_{j=1}^{n_{con}} c_j \{max[0, g_j(\mathbf{x})]\}^2 \tag{2.13}$$

where, $f(\mathbf{x})$ is the objective function, r_p is the penalty multiplier, n_{con} is the number of constraints, and c_j is the multiplier that may be needed to reflect different constraints. The penalty multiplier, r_p, is often a "large" number which can stay constant for the entire run or can vary from generation to generation. Other penalty functions for handling constraints include the exterior linear and the exterior step-linear penalty functions, expressed as follows:

$$\begin{aligned} P_j(\mathbf{x}) &= c_j\{max[0, g_j(\mathbf{x})]\} && \text{Exterior linear} \\ P_j(\mathbf{x}) &= \begin{cases} 0 & \text{if } g_j(\mathbf{x}) \leq 0 \\ c_j[1 + g_j(\mathbf{x})] & \text{else} \end{cases} && \text{Exterior step-linear} \end{aligned} \tag{2.14}$$

While a gradient-based approach for constrained problems must have first-order continuity (hence, the $\{max[0, g_j(\mathbf{x})]\}^2$ term in exterior quadratic penalty – commonly used by gradient-based approach), the GA does not have this requirement, so other forms – like the exterior linear and exterior step-linear penalty functions – are options.

The penalty addition technique to handle problem constraints is inefficient. Designs that are infeasible, but close to the constraint boundary, may contain important "genetic information" that is needed to find a feasible solution that is on the feasible side of the constraint boundary. Too strong a penalty may remove useful "genetic information" from the population, while too weak a penalty might not encourage feasible designs.

2.3.1.2 Design Variable Coding

In GA, an individual design in a population is often expressed in terms of binary-coded strings, known as chromosomes. The chromosomes comprise genes, where each

gene (an adjacent segment of 0s and 1s in the chromosome) represents a binary-coded design variable. The GA implemented in this work employs Gray coding to encode/decode these variables. The Gray code represents discretized values of a design variable within its upper and lower bound, where consecutive integers are represented by binary numbers differing in only one digit. Hence, all the design variables are discretized, including the continuous variables that are converted into a range of discretized values based on their resolution. The resolution, r_i, between discretized values of a continuous design variable, x_i, is expressed as follows:

$$r_i = \frac{x_i^U - x_i^L}{2^{b_i} - 1} \tag{2.15}$$

where, x_i^U is the upper bound on variable, x_i^L is the lower bound on variable, and b_i is the number of bits to code x_i. Ideally, for representing a continuous variable, the number of bits should be very large to make the resolution small, but this leads to an increase in the computational cost of the algorithm.

2.3.1.3 Selection Operator

The GA selection operator mimics the survival of the fittest approach by choosing which individuals out of the current population will become parents of the next generation of designs. The classical binary tournament selection technique is one of the most common selection operator, and is employed in this work.

The tournament selection puts the current generation individuals in an empty pot, called P1 for the discussion here. This technique randomly selects two individuals from P1 without replacement, and conducts a tournament that compares these two individuals based on their fitness function values. If the optimization problem intends to minimize the objective function, then the individual with lower fitness value (better design) is copied to the parent pool pot, P2. This process is repeated until P1 is empty, and the parent pool P2 is half full. The technique then refills the pot P1 with the current generation individuals and conducts a second tournament until the pot P1 is again empty. This way, every design in the current population competes

twice so that the best design gets two copies in the parent pool while the worst design is automatically rejected from further consideration. To keep the selection process simple, the parent pool has the size as the population size.

2.3.1.4 Crossover Operator

The crossover operator mimics the natural process of reproduction in which the genes from the parents are passed-on to the children. For computational simplicity, this work assumes that two parents form two children. A number of crossover techniques exist in the literature, such as: binary crossover, single point crossover, and multi-point crossover. Binary crossover has proven to be effective with the binary-coded GA and tournament selection approach, as suggested in Ref. [24]. The binary crossover technique transfers bits from a parent to a child based on a probability function. For uniform crossover, the first child receives the bit from the first parent with a 50% chance. This work employs uniform crossover technique to generate the next generation of points.

2.3.1.5 Mutation Operator

The mutation operator introduces new "genetic patterns" not present in the previous population. The probability of mutation is less than one percent. Williams and Crossley [24] derived the following empirical formula for mutation rate concerning binary-coded GAs using uniform crossover:

$$P_m = \frac{l+1}{2N_{pop}l} \tag{2.16}$$

where l is the length of chromosome, and N_{pop} is the population size. A high P_m value implies more exploration of the design space and an increased randomness in the search.

2.3.1.6 Elitism

Elitism is a technique to ensure that the best designs from the current generation pass directly to the next generation. Often this process is undertaken by replacing the worst individual designs in the current population by the best designs from the previous generations. This ensures that the best designs or the lowest fitness value (when minimizing an objective function) is not lost by mutation or design space exploration in the subsequent generations.

2.3.2 Multi-Objective Evolutionary Algorithms

As discussed in the previous section, single-objective EA finds only one near-global optimal solution. However, for a problem with multiple competing objectives, the ability of EA to explore the whole design space can lead to a Pareto-optimal set of solutions in one run of the algorithm. This capability of EA to find a set of trade-off designs as the generations progress makes it suitable for application to multi-objective problems. Several multi-objective GA approaches are available in literature, the earliest one being Vector Evaluated GA (VEGA), proposed by Schaffer [25] in 1985. A comprehensive list of several multi-objective evolutionary algorithms appears in Refs. [26–28]. Ref. [29] provides a detailed comparison of various multi-objective optimization algorithms.

Strength Pareto Evolutionary Algorithm (SPEA & SPEA-II) [5] is an elitist multi-criterion EA which implements elitism by maintaining a fixed number of designs in the non-dominated set. Non-Dominated Sorting Genetic Algorithm-II (NSGA-II) [30,31] is one of the most widely accepted multi-objective EA. This algorithm implements the idea of non-dominated sorting to evolve to a Pareto frontier. Basically, the combined parent and offspring population ($2N$) is divided into a number of non-dominated sets based on their level of dominance in the design space. The non-dominated sets are collected in a set of N designs based on their dominance, forming the parent pool for

the next generation. NSGA-II tends to suffer from a slow convergence rate after a few generations.

2.3.2.1 Two-Branch Tournament Genetic Algorithm

Crossley et al. [15] proposed the two-branch tournament GA with the motivation to compare designs on the basis of both the two competing objectives, rather than a single converted objective value. The overall process remains the same as the traditional GA with a modification in the tournament selection operator. The first branch of the selection operator assesses the individuals with respect to the first objective, while the second branch assesses the individuals with respect to the second objective. Fig. 2.2 shows the two-branch tournament selection GA using a flowchart.

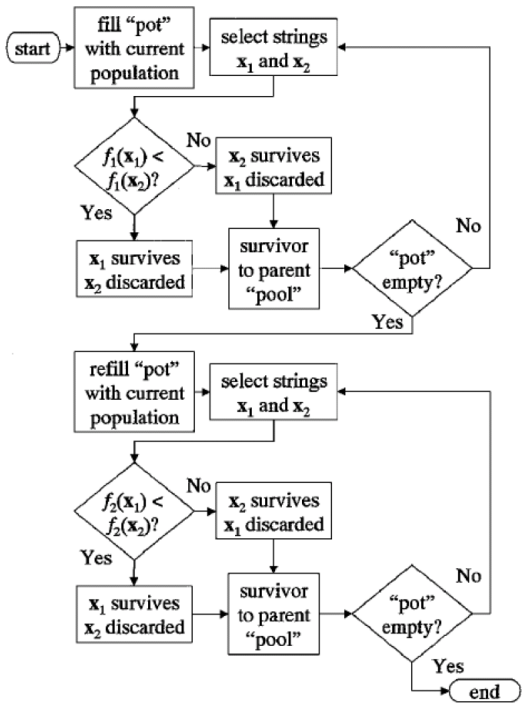

Figure 2.2. Flowchart for the two-branch tournament selection Genetic Algorithm [adapted from Ref. [15] (with permission)]

In this approach, the entire population is placed in a pot. Two individuals are randomly selected without replacement and compared on the basis of the first objective, ϕ_1. The better performing individual (having lower fitness value if the objective functions are minimized) is copied to the parent pool. This process is repeated until the pot is empty. At the end of the first tournament, the individuals in the parent pool are by nature strong in objective 1, or ϕ_1 strong. The pot is refilled and a similar tournament is conducted with respect to objective 2, adding ϕ_2 strong individuals to the parent pool. The crossover of two randomly selected parents hence results in 25% ϕ_1-ϕ_1 strong parents, 25% ϕ_2-ϕ_2 strong parents, and 50% ϕ_1-ϕ_2 strong parents. A modified two-branch tournament selection technique [3] is exploited as the global search optimizer for this thesis work. This modified approach is explained in detail in the next chapter.

2.4 Overview of Hybrid Optimization Approach

With the background discussion of gradient-based optimization techniques and population-based search algorithms presented in Sections 2.2 and 2.3, this section summarizes the need for and the limitations of the hybrid approach presented in Refs. [1–3] for constrained multi-objective MDNLP problems. This summary first discusses the benefit of creating the hybrid approach and then throws light on one of its major drawbacks, leading to proposal of the multi-fidelity approach developed for this thesis.

The motivation for the hybrid approach comes from the desire to mitigate the drawbacks of both the gradient-based optimization techniques and population-based search algorithms for solving constrained multi-objective MDNLP problems. As described in the previous sections, SQP – a gradient-based technique – cannot handle discrete design variables and tends to converge to a local optima because the underlying search works to satisfy conditions for a local optimum. However, SQP has the capability to enforce the constraints directly and strictly, and the solution obtained

by SQP can meet the conditions for local optimality. On the other hand, GA – a population-based search algorithm – cannot handle constraints directly and strictly. However, the GA can handle discrete variables while using its population search strategy to find a near-global optimum solution, but there is no associated way to show optimality of the solution. The combination of these algorithms allows SQP to handle the constraints and perform local optimization using the goal-attainment formulation, while the modified two-branch tournament GA performs population-based search and handles the discrete variables. The modified two-branch tournament GA can be extended to handle more than two-objectives, forming the N-branch tournament GA. As discussed back in Section 2.1, the two-branch tournament GA acts as the "overall" search strategy, and the fitness evaluation of each individual uses SQP to solve a goal attainment problem for the set of discrete variable values in the individual's chromosome, while using the continuous variable values in the chromosome as the initial point for the SQP. This leads to a set of non-dominated design solutions that meet local optimality conditions, and this set of designs represents the best trade-offs between both objectives in a two-objective problem. Hence, both of the algorithms can complement each other and improve the overall optimization process for constrained multi-objective MDNLP problems.

However, the hybrid approach, as employed in Refs. [1–3] is computationally expensive because each GA-level fitness evaluation requires a local optimization. If the SQP uses finite-difference derivatives, each of these local optimizations will require many function evaluations. So, for every individual in the GA population, the hybrid algorithm performs a local search. Then, if those function evaluations use "high-fidelity" analyses, the solution time for each of these fitness evaluation / local optimizations might be very high. The hybrid approach requires more function evaluations than the standalone GA itself. The standalone GA requires one "high-fidelity" analysis for each fitness evaluation, resulting in number of fitness evaluations equal to the number of individuals multiplied by the number of generations that the GA runs. Whereas, for the same problem, the hybrid approach will require a number of fitness

evaluations equal to the product of the number of individuals, the number of function evaluations conducted during local optimization of every individual, and the number of generations that the GA runs. The high computational time and cost involved with the hybrid approach can be demonstrated using an example. Assume that each "high-fidelity" function evaluation takes two minutes to complete, the population size for the modified two-branch tournament GA is set to 48, and the upper limit of function evaluations for SQP with goal-attainment formulation is set to 300 for every individual in the population. Hypothetically assuming that each individual requires an average of 150 function evaluations for the SQP search in every generation, and the GA terminates after 50 generations, the total number of function evaluations would be, $48*150*50 = 360{,}000$. This would result in a computational time of approximately 500 days (in serial computation) to generate a set of non-dominated solutions to this multi-objective problem, which clearly shows that the hybrid approach is inefficient for problems requiring analyses with modest computational cost.

The hybrid approach presents a promising technique to simultaneously address multiple competing objectives, nonlinear design space, and mix of categorically discrete and continuous design variables. These features compel the author to employ this hybrid technique as a backbone for the proposed multi-fidelity approach to solve constrained multi-objective MDNLP problems within a limited computational budget. To increase the efficiency of the hybrid approach, the author computationally assists the hybrid algorithm by utilizing "low-fidelity" function evaluations from approximation/surrogate models, reducing the computational cost involved in solving constrained multi-objective optimization problems.

2.5 Surrogate Modeling

Surrogate models are analytic models that approximate the input / output behavior of complex mathematical models or systems. The surrogate models are constructed from a finite set of actual calculations using the complex mathematical

model; these actual calculations are considered the "high-fidelity" analyses in the context of design optimization. A surrogate model approximates the output of a complex mathematical model at design points outside the limited set of "high-fidelity" points used to construct them. The purpose of a surrogate model is to reduce the computational time to perform an analysis; this reduction in computational time has an associated modeling error that can often make the surrogate less accurate than the original analysis. In simple words, surrogate modeling leads to the construction of an approximation model that tries to represent the response of simulation models for different design points. Because a surrogate is actually a model of a complex simulation model, it can also be termed as a metamodel [32].

There are two contexts for surrogate model construction. The first context is the global surrogate modeling in which the surrogate model represents the whole problem design space (i.e., the surrogate model predicts responses for any combination of variable values within the bounds of the problem). A global surrogate model generally sacrifices accuracy of prediction. Global surrogate models tend to provide "gross" representations of the functions, but with limited detail. A specific instance of this might be when a problem has multiple local minima that the surrogate model cannot reflect. The second context is the local surrogate modeling wherein the surrogate model represents only a specific region of the multi-modal design space. This local context might provide a higher accuracy through the ability to reflect more detailed behavior of the functions because the approximation is made over a very small range of the possible design variable values.

A number of surrogate modeling strategies exist; the literature describing these generally differentiate the strategies by the combination of basis functions deployed to construct the model [33, 34]. Most strategies follow the steps listed below [35]:

1. Identify the basis function.

2. Design an experiment or plan a sampling strategy to find design points to construct the surrogate.

3. Conduct "high-fidelity" analyses (simulation experiments) for a limited set of design points (known as the training points or sample points).

4. Construct the surrogate to fit the training data.

5. Assess the adequacy of the surrogate model (confidence intervals, hypothesis tests, lack of fit and other model diagnostics).

The surrogate model maps a computationally expensive function $y = f(\mathbf{x})$ with k design variables to an approximating function $\hat{y} = \hat{f}(\mathbf{x})$ using the "high-fidelity" function values for evaluations at n training points, where $\mathbf{x} \in \mathbb{D} \subset \mathbb{R}^k$. $\mathbb{D}$ denotes the design space, which is the domain defined by the design variables, k. Each i^{th} training point and its actual function value is denoted by $\mathbf{x}^{(i)} = (x_1^{(i)},...,x_k^{(i)})$ and $y^{(i)} = y(\mathbf{x}^{(i)})$ respectively. The approximating function $\hat{y}$ then cheaply predicts the value of the expensive black-box function at any desired point in the design space, D. Because $\hat{y}$ also includes modeling error, $\hat{y}$ provides a "low-fidelity" analysis for y.

The following section reviews different surrogate modeling strategies based on their basis functions. Although there are many surrogate modeling strategies, the section reviews only those two that are most relevant to this work – the interpolating surrogate models – Kriging models and RBF models.

2.5.1 Kriging Models

Kriging models are interpolating surrogate models that predict a more expensive or complicated function value at a specific point by computing a weighted average of the known values of the function in the neighborhood of the point. Danie G. Krige developed this approximation model for mining engineering with the motivation to estimate the most likely distribution of gold based on samples from a few boreholes [36]. Initial efforts for application of Kriging to computer experiments appears in Ref. [37]. Since then Kriging has been widely used to approximate nonlinear, computationally expensive objective and constraint functions in optimization problems [38–46].

Kriging models are a function of the spatial distance between design points, with their basis function given by the following mathematical expression:

$$\psi^{(i)} = exp(-\sum_{j=1}^{k} \theta_j |x_j^{(i)} - x_j|^{p_j}) \tag{2.17}$$

where $\mathbf{x}$ denotes the design points, k denotes the number of design variables, and θ_j and p_j are correlation parameters. θ_j is essentially a 'width' parameter that signifies how far a design point's influence extends, and the value of θ_j typically lies between 10^{-3} and 100 [33]. p_j acts like a 'smoothness' parameter, and its value varies between 0 and 2. As the design points get close to each other, the spatial distance between them tends to zero, causing the right hand side of Eq. 2.17 to approach a value of unity. Hence, the highest correlation occurs when the points are closest. Using a similar logic, points with a greatest distance from each other have the lowest correlation [33].

Kriging models provide function approximations by linearly combining a global trend function, μ, with a random process, $Z(\mathbf{x})$, given by the following expression [46]:

$$y(\mathbf{x}) = \mu + Z(\mathbf{x}) \tag{2.18}$$

where k is the number of basis functions, $y(\mathbf{x})$ is the unknown function value, and $Z(\mathbf{x})$ is a stochastic process with mean zero, variance σ^2, and non-zero covariance [33, 45–47].

The spatial correlation function for constructing Kriging models is given by the following mathematical expression:

$$\mathbf{R}[Z(\mathbf{x}^i), Z(\mathbf{x}^l)] = exp(-\sum_{j=1}^{k} \theta_j |x_j^{(i)} - x_j^{(l)}|^{p_j}) \tag{2.19}$$

For n sample design points, the correlation matrix $\mathbf{R}$ is a $n \times n$ matrix with its (i^{th}, j^{th}) element given by $\mathbf{R}[Z(\mathbf{x}^i), Z(\mathbf{x}^l)]$. The spatial correlation function shown in Eq. 2.17 and Eq. 2.19 corresponds to the general exponential function, which includes the exponential and Gaussian correlation functions. However, a number of different correlation functions exist, listed in Table 2.1 [48, 49]. This work employs the general exponential correlation function to build the Kriging models.

Table 2.1. Correlation functions for Kriging surrogate models.

Correlation Function	$R_j(\theta_j, x_1, x_2)$
Exponential	$exp(-\theta_j \lvert x_2 - x_1 \rvert)$
General Exponential	$exp(-\theta_j \lvert x_2 - x_1 \rvert^{p_j}),\ 0 < p_j < 2$
Gaussian	$exp(-\theta_j \lvert x_2 - x_1 \rvert^2)$
Linear	$max\{0, 1 - \theta_j \lvert x_2 - x_1 \rvert\}$
Spherical	$1 - 1.5\xi_j + 0.5\xi_j^3,\ \xi = min\{1, \theta_j \lvert x_2 - x_1 \rvert\}$
Cubic	$1 - 3\xi_j^2 + 2\xi_j^3,\ \xi = min\{1, \theta_j \lvert x_2 - x_1 \rvert\}$

To formulate the Kriging surrogate model, $2k + 2$ hyper-parameters need to be determined. These include k unknown θ_j values, k unknown p_j values, the σ^2 value, and the constant μ (leading to $2k+2$ unknown hyper-parameters). These parameters are estimated by maximizing the likelihood function given by the following expression:

$$L = \frac{1}{\sqrt{(2\pi\hat{\sigma^2})^n |\mathbf{R}|}} exp[-\frac{(\mathbf{y} - \mathbf{1}\hat{\mu})^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{1}\hat{\mu})}{2\hat{\sigma^2}}] \tag{2.20}$$

where $\mathbf{1}$ denotes a k-dimensional unit vector, and, $\hat{\mu}$ and $\hat{\sigma^2}$ denote the maximum likelihood estimates for μ and σ^2 respectively.

The maximum likelihood estimates for $\hat{\mu}$ and $\hat{\sigma^2}$ are obtained by taking partial derivatives (with respect to $\hat{\mu}$ and $\hat{\sigma^2}$) of the natural logarithm of the likelihood function (log-likelihood function) given in Eq. 2.20. Equating these two expressions to zero gives,

$$\hat{\mu} = \frac{\mathbf{1}^T \mathbf{R}^{-1} \mathbf{y}}{\mathbf{1}^T \mathbf{R}^{-1} \mathbf{1}} \tag{2.21}$$

$$\hat{\sigma^2} = \frac{(\mathbf{y} - \mathbf{1}\mu)^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{1}\mu)}{n} \tag{2.22}$$

Substituting the values of $\hat{\mu}$ and $\hat{\sigma^2}$ obtained in Eq. 2.21 and 2.22 into Eq. 2.20, the likelihood function depends only on θ_j and p_j. These hyper-parameter values are obtained by solving a $2k$ dimensional unconstrained non-linear optimization problem; maximizing the likelihood function with θ_j and p_j as decision variables. This

NLP problem can be solved by using either evolutionary algorithms (such as GA or Simulated Annealing) or using gradient-based algorithms (such as SQP or Newton-Raphson method). Because this is a global optimization problem with many possible local optima, using a gradient-based search requires a multi-start approach to find the hyper-parameters. The θ_j and p_j values from this log-likelihood problem are used to evaluate $\mathbf{R}$, μ, and σ^2. The following expression gives the linear estimator to predict the function value, $\hat{y}$, at any point $\mathbf{x}$ using the Kriging model.

$$\hat{y}(\mathbf{x}) = \hat{\mu} + \mathbf{r}^T(\mathbf{x})\mathbf{R}^{-1}(\mathbf{y} - \mathbf{1}\hat{\mu}) \tag{2.23}$$

where $\hat{y}$ is the predicted function value at design point $\mathbf{x}$, and $\mathbf{r}^T$ is the correlation vector between $\mathbf{x}$ and the sampled design points $\{x_1,...,x_n,\}$, expressed as:

$$\mathbf{r}^T(\mathbf{x}) = [\mathbf{R}(\mathbf{x}, x_1), ..., \mathbf{R}(\mathbf{x}, x_n)]^T \tag{2.24}$$

Forrester et al. [33] recommend Kriging as a good surrogate modeling strategy for problems having less than 20 design variables (k) with a sample size (n) limited to 500 points. Although, Kriging models show great promise for building accurate global approximations of a design space [37], the additional effort required to solve for the hyper-parameters increases the computational time required to construct these approximation models, making them computationally expensive when compared to simple response surface models.

This work initially investigated Kriging models as a surrogate modeling strategy to approximate function / constraint values to solve MDNLP problems using proposed the multi-fidelity hybrid approach. Later, this work replaces the Kriging models with simpler (and, hence, comparatively computationally cheaper) RBF models, without altering any other aspect of the multi-fidelity hybrid approach.

2.5.2 Radial Basis Function Models

The RBF method is one of the primary tools to develop approximation models for multidimensional scattered data. Rolland Hardy [50] developed this approximation

method for modeling irregular topographical contours of geographical data. The past two decades have witnessed an increasing interest in this simple yet effective modeling strategy [51–60].

RBF models are real-valued approximation functions whose values depend on their distance from the origin or a certain point $\mathbf{c}$, known as a center. The center is a subset of the "high-fidelity" design points that act as the training points (or sample points) for making the RBF model. The RBF approximation ($\hat{f}$) is given by the following mathematical expression [33]:

$$\hat{f}(\mathbf{x}^{(j)}) = \mathbf{w}^T \Phi = \sum_{i=1}^{n_c} w_i \phi(||\mathbf{x}^{(j)} - \mathbf{c}^{(i)}||) = y^{(j)} \tag{2.25}$$

where $j = \{1,...n\}$, $\mathbf{w}$ denotes the basis weights, $\mathbf{c}^{(i)}$ denotes the i^{th} of the n_c basis centers, and Φ contains the values of the basis functions ϕ evaluated at Euclidean distance between the sample design point $\mathbf{x}$ and the basis center $\mathbf{c}^{(i)}$. Although Eq. 2.25 shows that the predictor, $\hat{f}$, is linear in terms of basis function weights $\mathbf{w}$, it can still predict highly nonlinear function values. Hence, RBF modeling strategy can be implemented to model nonlinear design space containing multiple local minima / maxima.

The approximation strategy shown in Eq. 2.25 is similar to the artificial neural network approach. This formulation represents a single-layer neural network with radial coordinate neurons, having an input $\mathbf{x}$, hidden units Φ, weights $\mathbf{w}$, linear output transfer functions and output $\hat{f}(\mathbf{x})$ [33].

This work uses RBF models with a series of basis functions that are symmetric and centered at each sample point, which means that the basis center actually coincides with the sample points – $\mathbf{c}^{(i)} = \mathbf{x}^{(i)}$, where $\mathbf{x} = \{\mathbf{x}^1,...,\mathbf{x}^n\}$. This leads to Eq. 2.26 [33]:

$$\Phi \mathbf{w} = \mathbf{y} \tag{2.26}$$

where Φ denotes the Gram matrix, and $\mathbf{y} = \{y^1,...,y^n\}$ denotes the known function values at the sample points $\mathbf{x}$. The Gram matrix is defined as follows:

$$\Phi_{i,j} = \phi(||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||) \tag{2.27}$$

Table 2.2. Common basis functions for RBF modeling.

Type of Basis Function	$\phi(r)$
Linear	r
Cubic	r^3
Thin plate spline	$r^2 \log r$
Gaussian	$e^{-r^2/(2\sigma^2)}$
Multiquadric	$(r^2+\sigma^2)^{1/2}$
Inverse multiquadric	$(r^2+\sigma^2)^{-1/2}$

with $i,j = \{1,...n\}$. The weights, $\mathbf{w}$, are calculated using $\mathbf{w} = \Phi^{-1}\mathbf{y}$. However, the choice of the basis function ϕ affects the calculation of the weights.

The various basis functions for modeling RBF approximations appear in Table 2.2. Linear, cubic and thin-plate spline basis functions are known as fixed basis functions. For these functions, the number of parameters to be estimated for RBF approximation stands at one for each basis function. Gaussian, multiquadric and inverse multiquadric basis functions are known as parametric basis functions. These basis functions represent the nonlinear design space much better than the fixed basis functions at the expense of a more complex parameter estimation process. This involves the estimation of σ as an extra parameter introduced via the parametric basis functions along with estimating the weights, $\mathbf{w}$. Usually the value of σ is taken to be same for all basis centers, increasing the generalization of the RBF model at the expense of estimating just one extra parameter as compared to the fixed basis functions. Taking a closer look at the Gaussian basis functions, choosing different σ values for each basis center eventually leads to the Kriging basis function given in Eq. 2.17. Hence, the RBF Gaussian basis function is actually a simplified version of the Kriging basis function, with fewer parameters to be solved for RBF Gaussian modeling leading to a lower computational burden when compared to Kriging surrogate modeling.

The Gaussian and multiquadric functions are one of the most popular basis functions among those listed in Table 2.2. The reason for their popularity can be attributed to the their ability to generate a positive definite symmetric Gram matrix while computing the weights, $\mathbf{w}$, using Cholesky factorization [61]. Cholesky factorization is a technique that decomposes a positive definite symmetric matrix into an upper triangular matrix and its transpose, used for numerical calculation of the inverse of a matrix. However, very close proximity of two sample points can lead to ill-conditioning of the Gram matrix, because the distance between $x^{(i)}$ and $x^{(j)}$ is so small that $\Phi_{i,j}$ approaches zero, causing the Cholesky factorization to fail. This issue will not cause any problems when a space-filling sampling plan is used for the sampling points, because a space-filling plan intentionally scatters sample points throughout the design space. However, if clusters of sampling points focusing on a specific region of the design space are used, RBF modeling could fail.

This works uses RBF modeling with a provision to ensure that the sampling design points do not get too close to each other, ensuring that RBF modeling does not fail. The motivation to integrate RBF with the multi-fidelity hybrid approach for the latter part of this work comes from the lower number of parameters that must be estimated when constructing RBF models.

2.5.3 Sampling Strategies

For generating a global surrogate model, it is important to use sampling points that exhibit good coverage of the whole design space. A good sampling strategy works to find these points, ultimately leading to the formation of a good surrogate model that better represents the design space. Because the accuracy of approximation is better at a design point close to one of the sample points than at a point further away from them, it would be beneficial to have a sampling plan that spreads the sample points uniformly throughout the design space (space-filling sampling plan). Some of the methods to create a uniform spread of the space-filling include full-factorial

sampling, Latin-Hypercube Sampling (LHS), Monte-Carlo methods, orthogonal arrays [33]. Based on the work in Ref. [62], this work employs LHS technique to generate the initial set of sampling points (replacing the randomly-generated initial population in the hybrid approach [1–3] with an LHS approach in the multi-fidelity approach presented in this work), so the next subsection describes this.

2.5.3.1 Latin-Hypercube Sampling

LHS technique works by partitioning the design space into equal sized hypercubes and placing a point in each hypercube, ensuring that if the design space is exited along any direction parallel to any axes, no other filled hypercube would be encountered. This work uses MATLAB's Statistics Toolbox [63] to generate the LHS sampling. The LHS design in the toolbox is based on the "maximin" criterion, developed by Johnson et al. [64]. This criterion tends to generate a space-filling sampling by maximizing the minimum distance between all the sample points.

2.6 Surrogate-Based Optimization

The motivation behind surrogate-based optimization is to limit the number of "high-fidelity" function evaluations (computationally expensive actual function evaluations) to as few as possible. These optimization techniques use surrogate models to approximate function / constraint values with much faster to evaluate functions, which leads to a significant reduction in the computational time required to solve an optimization problem. This reduction in computational time, and hence cost, can be directly attributed to the reduction in the number of "high-fidelity" function evaluations required by these surrogate-based optimization techniques.

Surrogate-based optimization has recently gained popularity for solving problems with expensive objective / constraint functions [55, 56, 65, 66]. Examples of surrogate-based optimization to complex engineering problems appear in Refs. [33, 52]

Jones et al. [38] developed the popular surrogate-based optimization technique – Efficient Global Optimization (EGO) – which combines a Bayesian approach with Kriging surrogate modeling to reduce the function evaluations for global search, while handling only continuous variables.

Several new surrogate-based optimization techniques that can handle mixed integer programming problems have appeared recently; see, for example, Refs. [59, 60, 67, 68]. One of these techniques is Surrogate Optimization-Mixed Integer (SO-MI) [59], which uses a stochastic sampling approach to determine new training points (also know as infill points) for constructing (or updating) the surrogate models. This technique uses cubic RBF models, while implementing a branch-and-bound approach to handle the integer variables. An updated version of this approach is Mixed-Integer Surrogate Optimization (MISO) [60], which uses several sampling strategies to determine the infill points for updating the RBF models. Very recently, Muller and Woodbury developed Global Optimization with Surrogate Approximation of Constraints (GOSAC) [69], which uses a two-phase optimization approach with RBF models to solve mixed-integer problems with computationally cheap objective function and expensive black-box constraints. Roy [70–72] developed a mixed-integer EGO framework employing Kriging models for large-scale problems and leverages the computationally efficient framework of NASA's Open source Multidisciplinary Design Analysis and Optimization (OpenMDAO) to solve a simultaneous aircraft design, airline allocation and revenue management problem.

To handle both continuous and discrete variables, Kolencherry [62, 73] combined surrogate modeling techniques with a single objective binary-coded genetic algorithm to find optimum design solutions using sequential Kriging surrogate modeling. This technique approximated function values using Kriging models and selectively replaced the design point having the worst "low-fidelity" fitness with the design point having the best "low-fidelity" fitness. A "high-fidelity" function evaluation of the best point was conducted, and this point was used to update the Kriging model in the next iteration. This reduced the number of "high-fidelity" runs required by the GA for

global optimization. However, this approach suffers from the same constraint handling issues as a traditional GA and uses an exterior penalty function to reflect constraints in the fitness function.

The key to obtaining a global solution using surrogate-based approach lies in the balance between exploiting the surrogate (local search near the expected minima where the prediction error of objective / constraint values using the surrogate is expected to be low) and exploring the design space (global search where prediction error of objective and / or constraint values using the surrogate is expected to be high). This work leverages surrogates for conducting the local search portion of the multi-fidelity hybrid algorithm, while relying on "high-fidelity" analyses for the global search. This reduces the number of "high-fidelity" function evaluations associated with the hybrid algorithm, making it more efficient to solve complex engineering problems.

3. MULTI-FIDELITY APPROACH FOR CONSTRAINED MULTI-OBJECTIVE MIXED DISCRETE NONLINEAR PROGRAMMING PROBLEMS

As discussed in the previous chapter, the standalone hybrid approach [1–3] can solve constrained multi-objective MDNLP problems by combining global search using an evolutionary algorithm with a gradient-based local search technique. However, using a gradient-based approach that requires "high-fidelity" models for evaluating objective(s) and / or constraints is computationally expensive. To reduce the computational cost of the standalone hybrid approach, this work exploits surrogate modeling techniques.

3.1 Methods and Approach

The proposed multi-fidelity hybrid algorithm combines the modified two-branch tournament GA for global search with surrogate-assisted local search using the goal attainment SQP algorithm provided in the *fgoalattain* solver available in the MATLAB Optimization Toolbox [74]. The problem statement for the proposed algorithm contains three components – surrogate models, two-branch tournament genetic algorithm, and sequential quadratic programming. These components are explained in detail in the following sections.

3.1.1 Surrogate Models

The multi-fidelity approach first builds global surrogate models for each objective and constraint function in the problem, providing "low-fidelity" approximations for all objective and constraint functions during the local search phase of the algorithm.

The LHS technique generates the initial population for the two-branch tournament GA. This initial population acts as a sample set for constructing the initial global surrogate models. The LHS generated initial population is different than the more traditional, randomly generated initial population. The sample set and hence, the surrogate models, are updated every generation by selectively adding design points obtained after the local search conducted by the SQP algorithm. The sample set for constructing the surrogate models is always scaled between 0 and 1, as recommended by Forrester et al. in Ref. [33]. This means that the smallest design variable value (lower bound of design variable) corresponds to zero, the largest design variable value (upper bound of design variable) corresponds to 1; almost always via a linear transformation.

The combination of GA and SQP works to to converge the GA population to a representation of the Pareto frontier. With every subsequent GA generation, the GA population converges along the Pareto frontier and after several generations, the population tends to focus along this frontier. This leads to the issue where design points or individuals in the GA population might be very close to the design points present in the sample set for constructing the surrogate models, while still being spread out in the design space that corresponds to Pareto optimal designs. Design points with very close proximity to each other (or clusters of points in a specific region of the design space) tend to cause matrix ill-conditioning issues – leading to the failure of Cholesky factorization while building surrogate models, as mentioned in Section 2.5.2. An acceptance criterion works to prevent this matrix ill-conditioning by accepting new points for inclusion in the sample set on the basis of their spatial distance from the points already in the sample set. This acceptance criterion is explained in detail in Section 3.2.1.

This thesis research initially investigated Kriging as the surrogate modeling strategy and thereafter used RBF for surrogate modeling due to the inherent benefit that the process of constructing RBF surrogate models is computationally cheaper than constructing Kriging models (discussed in Section 2.5.1). However, the multi-fidelity

hybrid algorithm framework can use almost any (or maybe any) surrogate modeling strategy for approximations in the local search / fitness evaluation.

3.1.1.1 Kriging Models

In this research, the Kriging models are based on the Kriging toolbox setup of Forrester at al. [33]. However, two changes have been embedded in this setup.

- First, to find the optimum value of the correlation function parameters, θ_j, this work employs a multi-start approach using MATLAB's constrained non-linear optimization solver *fmincon* [75], with 12 different sets of initial θ_j values, typically varying from 10^{-3} to 100 [33], generated using the LHS method. The reason for performing twelve initial starting values of θ_j (instead of any other random number) lies in the fact that the server used for conducting the runs connects to 12 parallel processors at once. Hence, performing 12 *fmincon* runs in parallel took similar time as compared to a single *fmincon* run. In this manner, the multi-start local search helps find very good values for θ_j without the very high expense of conducting a non-gradient global search for θ_j. Reference [49] implements a similar scheme to find the correlation function parameters.

- Second, the Kriging basis function in this setup uses a general exponential correlation function with a small modification – the correlation parameter p_j is same for all the design variables, essentially making it scalar p ranging from 0 to 2 (refer to Table 2.1). Reference [76] shows a comparison of the various Kriging correlation functions possible. The parameter p is an additional variable along with the θ_j values found using the *fmincon* solver to maximize the log-likelihood function.

3.1.1.2 Radial Basis Function Models

The RBF models implemented in this work are based on the SURROGATES toolbox setup [77], using the RBF models by Jekabsons [78]. This work employs Gaussian basis functions for constructing the RBF models. The extra parameter, σ, is not calculated using an optimization problem, instead, the work here uses a fixed value throughout the entire run of the multi-fidelity framework to solve the constrained multi-objective MDNLP problem. Using an approach that keeps σ fixed for the entire helps to reduce the computational time required to construct and update the RBF models.

Estimating σ Value for RBF models: For this work, the σ value for constructing the RBF models is estimated by using a directed trial-and-error approach. This involves computing the prediction error of the objective and constraint RBF models by implementing the Leave One Out Cross Validation (LOOCV) technique for different σ values. The author recommends using 0.1, 0.5, 1.0, and 1.5 as the initial trial σ values for this approach.

The LOOCV technique tests the accuracy of a surrogate model by separating the sample set into two parts. For n points in a sample set, LOOCV uses $n - 1$ points to construct a surrogate model and then calculates a prediction error for the remaining point using the same surrogate model. This process is repeated until every point in the sample set is left out once from forming a surrogate model and a prediction error is computed for the left out sample point by using the surrogate predicted value and its actual "high-fidelity" function value. The accuracy of the RBF model is then quantified by calculating its Root Mean Square (RMS) error value using the predicted errors obtained for every sample point.

The author recommends running the multi-fidelity hybrid algorithm for three GA generations using each of the initial trial σ values. This trial-and-error approach calculates the RMS error values for all the objective and constraint RBF models after the completion of the third GA generation. These error values obtained for all the

RBF models using different σ values are then compared to identify the σ value that leads to minimum prediction error (given by the minimum RMS error value) for all (or most) of the objective and constraint RBF models. This σ value is considered to be suitable for constructing the RBF models for a particular problem. Usually the σ value is problem dependent and tends to vary from problem to problem. The author recommends searching for a more suitable σ value by conducting the same trial-and-error approach for some more integer values close to the chosen σ value. The value of σ stays constant throughout the algorithm run for every problem. A study of the change in prediction error of the RBF models by using the same σ value throughout the optimization run appears in Section 4.1.2.3. This is a somewhat ad hoc approach, but it provides a computationally efficient approach that – for the problems solved here – provides good results.

3.1.2 Two-Branch Tournament Genetic Algorithm

The two-branch tournament GA solves an unconstrained multi-objective optimization problem with both the continuous and discrete variables. The multi-objective GA already formulated in the standalone hybrid approach [1–3] employs a modified two branch tournament selection technique, along with the mutation and crossover operators. The design variables represented in the chromosome of every individual in the GA act as starting points for the local search to evaluate that individual's fitness function. However, only the continuous variables ($\mathbf{x}_c$) undergo minimization using the SQP algorithm and the discrete variables remain constant throughout this goal-attainment search. The formulation for two-branch tournament GA appears below:

Minimize:

$$\begin{aligned} &f_1(\mathbf{x}_d, \mathbf{x}_c^0) \\ &f_2(\mathbf{x}_d, \mathbf{x}_c^0) \end{aligned} \tag{3.1}$$

Subject to:

$$\begin{gathered}(x_c^0)_i^L \leq x_i \leq (x_c^0)_i^U \\ (x_d)_i \in A, B, C, D, \ \text{... (Discrete variables)}\end{gathered} \tag{3.2}$$

where, $\mathbf{x}_c^0$ is the vector of initial continuous variable values (starting point) used in the local search, and $\mathbf{x}_d$ is the set of discrete variable values that remain constant throughout the SQP optimization.

3.1.2.1 Modified Two-branch Tournament Selection

The modified two-branch tournament selection employed in the standalone hybrid approach [1–3] divides the parents into three sub-pools. This is in addition to the two parent pools (based on the fitness function associated with the first objective, ϕ_1, and second objective, ϕ_2) already created in the original two-branch tournament selection approach [15] for crossover (refer to Section 2.3.2). The first sub-pool contains ϕ_1-ϕ_1 type parents, the second sub-pool contains ϕ_2-ϕ_2 type parents, and the third sub-pool contains ϕ_1-ϕ_2 type parents, all paired for crossover operation within their respective sub-pools. The modified two-branch tournament requires the population size to always be a multiple of 8, i.e., $8n$, to enable the formation of sub-pools as explained in the following example.

Fig. 3.1 illustrates the modified tournament selection technique with an example. Consider the population size to be 8 ($n = 1$). After the two-branch tournament selection process, 4 parents are ϕ_1-strong (i.e., they were selected based upon their performance in the first objective function) and the other 4 are ϕ_2-strong (similarly, selected based upon their performance on the second objective function), divided into separate parent pools. The modified tournament selection further divides the parent pool into sub-pools using selective parent mixing. Half of the parents from pool 1 are randomly moved to sub-pool 1, creating a mix of ϕ_1-ϕ_1 type parents that would lead to ϕ_1-strong offspring after crossover. Similarly, half of the parents from pool 2 are randomly moved to sub-pool 2, creating a mix of ϕ_2-ϕ_2 type parents that would

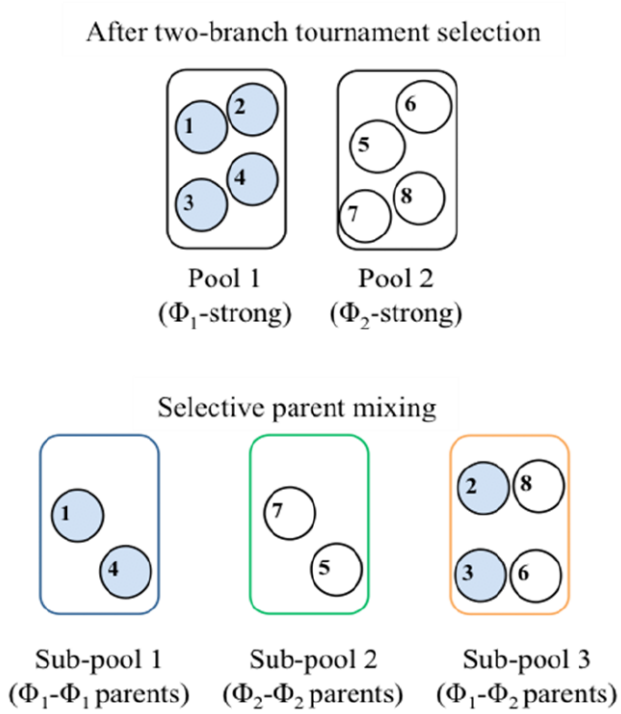

Figure 3.1. Illustration of the modified two-branch tournament selection [adapted from Ref. [1, 3] (with permission)].

lead to ϕ_2-strong offspring after crossover. The remaining parent population from both parent pools is moved to sub-pool 3, allowing crossover between ϕ_1-strong and ϕ_2-strong parents.

3.1.3 Goal Programming via Sequential Quadratic Programming

The SQP algorithm searches the continuous design space using the goal-attainment formulation to find non-dominated trade-off designs that represent a Pareto frontier, similar to the standalone hybrid approach [1–3]. The *fgoalattain* solver converts the multi-objective problem into a single-objective optimization problem by converting all the objectives into a set of inequality constraints and minimizes a slack variable, γ, as the objective. The goal values for every individual in the population, f_l^G, are

generated by using the goal formulation technique presented under the next heading. The goal-attainment problem formulation solved via SQP appears below:

Minimize:

$$\gamma \tag{3.3}$$

Subject to:

$$\hat{f}_l(\mathbf{x}_c) - \alpha_l \gamma \leq f_l^G \qquad (l = 1, 2) \tag{3.4}$$

$$\hat{g}_j(\mathbf{x}_c) \leq 0 \tag{3.5}$$

$$\hat{h}_k(\mathbf{x}_c) = 0 \tag{3.6}$$

$$(x_c)_i^L \leq (x_c)_i \leq (x_c)_i^U \tag{3.7}$$

The weights, α_l, are set as absolute values of the corresponding goal values, f_l^G, as discussed in Ref. [1–3]. The multi-fidelity approach presented here uses the surrogate models to provide "low-fidelity" objective and constraint approximations to *fgoalattain*, which uses them to find the optimized continuous design variables, $\mathbf{x}_c^*$, for every individual in the GA population (while keeping the discrete design variables fixed for every individual in this step). These optimized designs so obtained satisfy all "low-fidelity" constraints but may or may not satisfy the actual "high-fidelity" problem constraints. This is different from the standalone hybrid approach which uses actual ("high-fidelity") objective and constraint function values to conduct this local search step using goal-attainment via SQP, providing constraint satisfaction as part of the local optimality. However, the multi-fidelity approach here gives up some of this benefit of the standalone hybrid approach with the intent of greatly reducing the computational cost.

This SQP goal-attainment approach (using *fgoalattain* solver) seems to be successful in enforcing "low-fidelity" problem constraints (approximated from the surrogate models for constraint functions), except for two cases when the SQP algorithm would not be able to successfully handle these problem constraints. First case, when SQP is unable to find a feasible local solution for a given starting point. Second case, when

SQP is unable to find a feasible solution within the maximum possible number of iterations, set to MATLAB's default value for this work. In both these cases, the designs receive a high objective value penalty in the GA-level problem in an effort to remove them from the subsequent GA generations.

3.1.3.1 Goal Formulation Technique

This work employs the goal formulation technique developed in the standalone hybrid approach [1, 3]. The goal formulation technique assigns goal values to every individual in the population based on the sub-pool it belongs to among those generated in the modified two-branch tournament selection (refer to Section 3.1.2.1). The *fgoalattain* solver employs these goal values to perform the goal-oriented local search while satisfying the problem constraints. The hybrid approach identifies an ideal point – a combination of minimum f_1 and minimum f_2 values – to find the utopia point. The utopia point is set as 0.9 times the ideal point. If a lower f_1 and (or) f_2 value is available in the subsequent GA generations, the ideal point changes, also updating the the utopia point. For two-objective problems, a set of perpendicular lines originate from the utopia point, shown in Fig. 3.2 as dashed lines. The point of intersection of a goal vector – originating from an individual in the population – with the dashed lines defines the goal point for that individual.

In this goal formulation technique, children of parents from sub-pool 1 are assigned a goal vector with a zero slope that tends to seek maximum improvement along the direction of objective 1, f_1. Children of parents from sub-pool 2 are assigned a goal vector with a 90 degree slope that tends to seek maximum improvement along the direction of objective 2, f_2. However, for children with parents from sub-pool 3, the goal vector depends on their relative spatial position in the design space, with an individual having better objective 1 value inclined towards more improvement in objective 1 and an individual having better objective 2 value inclined towards more improvement in objective 2.

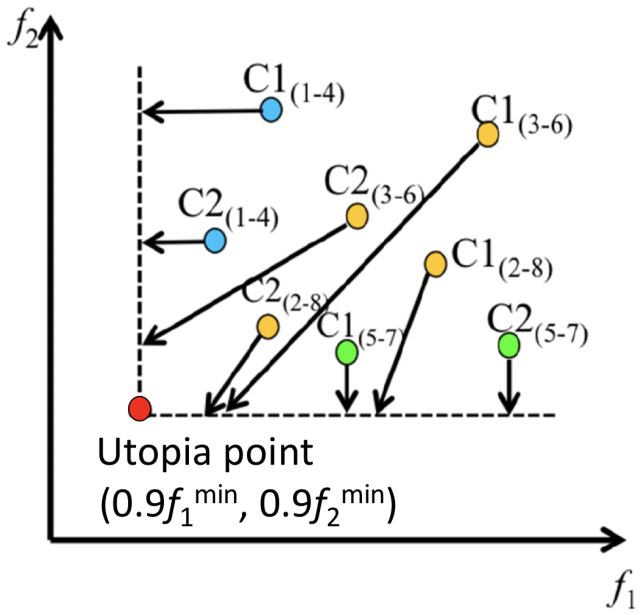

Figure 3.2. Illustration of the goal formulation technique [adapted from Ref. [1,3] (with permission)].

Fig. 3.2 illustrates the goal formulation technique following the example presented in Fig. 3.1. Parents 1 and 4 from sub-pool 1 create offspring $C1_{1-4}$ and $C2_{1-4}$. These individuals are assigned goal points that seek to improve their f_1 values, without improving their f_2 values. Similarly, $C1_{5-7}$ and $C2_{5-7}$ result from sub-pool 2 and are assigned goal points that seek to improve their f_2 values, without improving their f_1 values. $C1_{3-6}$, $C2_{3-6}$, $C1_{2-8}$, $C2_{2-8}$, all result from sub-pool 3 and are assigned goal points that seek to improve both their f_1 and f_2 values.

3.2 Multi-Fidelity Optimization Framework

The LHS strategy generates initial design points at the beginning of the two-branch tournament GA. The GA population size (number of initial design points for GA) depends on the number of design variables present in the problem in consider-

ation. Section 3.2.3 provides a technique to approximate the GA population size for different problems.

3.2.1 Algorithm Description

Fig. 3.3 shows a simplified flowchart depicting the communication between the GA, SQP and the surrogate models. The left portion of the figure signifies the basic framework of the multi-fidelity hybrid algorithm, which possesses similar basic operational characteristics as a GA. The right portion of the figure depicts the local optimization that is conducted to evaluate the fitness function values of every individual in the GA population. Once again it is pointed out that the basic algorithm remains the same for both Kriging and RBF surrogate modeling strategies. A detailed description of the steps involved in the proposed mutli-fidelity hybrid algorithm appears below. For algorithm description purposes, the term "high-fidelity" will be referred to as *hifi*, and the term "low-fidelity" will be referred to as *lofi*.

- GENERATE THE INITIAL POINTS: The algorithm generates a set of initial design points (initial population for the GA) using the LHS strategy.

- EVALUATE THE INITIAL (SAMPLE) POINTS: This step evaluates the initial design points (sample set for surrogate models) using *hifi* analysis to find their actual objective and constraint values. All the design points are scaled between 0 and 1 using the upper and lower bounds of each design variable.

- CREATE *hifi* DATABASE: To ensure that no design point undergoes *hifi* analysis more than once, a database stores the objective and constraint information for these design points.

- CONSTRUCT SURROGATE MODELS: The points with *hifi* evaluations (the initial GA population here) act as sample points to construct surrogate models for each objective and constraint function in the problem. These models give

approximate objective and constraint information for evaluating different design points.

- MODIFIED TWO-BRANCH TOURNAMENT SELECTION: The GA population is ultimately divided into three sub-pools based on the fitness values of the individuals with respect to the first and the second objective.
- GOAL ASSIGNMENT: The goal formulation technique assigns a goal point to every individual in the GA population.

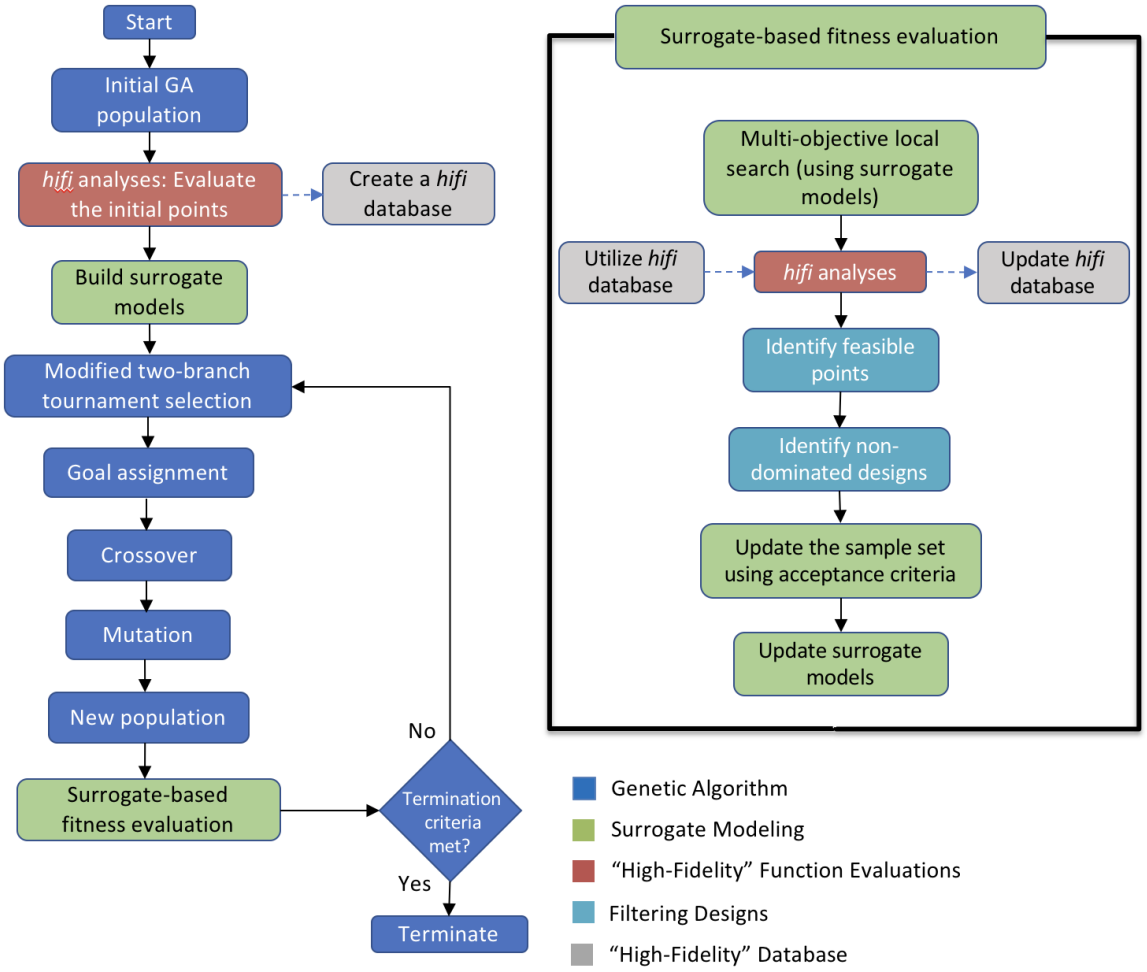

Figure 3.3. A simplified flowchart depicting the multi-fidelity multi-objective hybrid algorithm.

- CROSSOVER & MUTATION: The crossover occurs between the respective sub-pools (generated by modified tournament selection operator), followed by the mutation operator.
- NEW POPULATION: A new GA population is generated which now requires local search to evaluate the fitness function values for all the individuals in the new GA population.
- SURROGATE-BASED FITNESS EVALUATION: The right section of Fig. 3.3 depicts the local search step that is undertaken using the *lofi* approximations provided by the objective and constraint surrogate models.
  - MULTI-OBJECTIVE LOCAL SEARCH: The SQP algorithm with goal-attainment formulation employs the surrogate models to perform local search using MATLAB's *fgoalattain* solver, with the individual goals assigned in the previous step.

    This step performs gradient-based optimization by using the objective and constraint surrogates, instead of conducting actual (*hifi*) function / constraint evaluations. The local search works to find objective values nearest to the assigned goal values while satisfying the *lofi* problem constraints for every individual in the GA population. If the local search encounters any infeasible point for which the objectives cannot be minimized while satisfying the problem constraints, it assigns very high objective values to that design point as a penalty.
  - *hifi* ANALYSES: This step conducts *hifi* analyses for each of the individuals that satisfy the *lofi* constraints. These individuals are referred to as *new_hifi* for algorithm description purposes. At this step, the algorithm communicates with the database to ensure that *hifi* information for design points that are already existing in the database is not re-calculated.
  - UPDATE THE DATABASE: With every generation, the algorithm updates the database by adding *hifi* information for any new design point,

making the database grow dynamically. However, a database like this could grow to be very large. Hence, caution needs to be exercised in this database looking approach, because for some problems, at some point, the computational cost to sort through the list might make this approach irrelevant.

– IDENTIFY FEASIBLE POINTS: Using the *hifi* information obtained in the previous steps, the algorithm performs a constraint check to find feasible points – among the *new_hifi* points – that satisfy the actual (*hifi*) problem constraints. These individuals are referred to as *feas_hifi* points for algorithm description purposes.

  It is possible that the *new_hifi* points that satisfy the *lofi* constraints may not satisfy the actual (*hifi*) problem constraints, so they cannot be used to find a reliable non-dominated set of designs that will lead to a Pareto frontier between the two competing objectives.

– IDENTIFY NON-DOMINATED DESIGNS: The *feas_hifi* points from the previous step compete for inclusion in the non-dominated design set. The points from the non-dominated set represent the Pareto frontier.

– UPDATE THE SAMPLE SET: The algorithm employs an acceptance criterion to filter and add points from the *new_hifi* set to the already existing set of *hifi* points. As mentioned before, all the design points are scaled between 0 and 1 using the upper and lower bounds of each design variable.

  Acceptance Criterion: The acceptance criterion uses the spatial distance of the concerned point from all the *hifi* points as a basis for selecting points into the sample set. This approach constructs imaginary hyper-spheres with fixed radius, R, for every *hifi* point, with each point acting as the center for their respective hyper-spheres. The algorithm calculates the spatial distance of each *new_hifi* point from all the *hifi* points, and checks whether that point lies outside all hyper-spheres. If true, then the point

gets added to the *hifi* set of points, ensuring that the next point in *new_hifi* set calculates its distance from the original *hifi* set of points and the newly added point. All the design points are scaled between 0 and 1 to ensure that there are no scaling issues while comparing different design points. This makes the *hifi* set grow dynamically with every new selection from *new_hifi* set of points, leading to the employment of an equal or increased number of sample design points for constructing each objective and constraint surrogate model, when compared to the surrogate models in the previous generation. A value of $R = 0.05$ seems to work for all problems tested here. This, too, is a somewhat ad hoc selection made by some trial and error while using the approach to solve test problems.

  - UPDATE THE SURROGATE MODELS: The updated set of *hifi* points act as the sample set to update the surrogate models.

- CONTINUE WITH ALGORITHM: The algorithm then continues with the modified two-branch tournament selection, the goal assignment, crossover, and mutation operations to generate the new GA population. The algorithm continues with the steps as described above until any termination criteria is satisfied.

3.2.2 Termination Criteria

This work employs three different termination criteria to terminate the algorithm. These termination criteria prevent any wastage of "high-fidelity" function evaluations by stopping the algorithm after a fixed number of GA generations or when there is no alteration in the non-dominated set for a few consecutive generations. Out of the three criteria, whichever criterion gets satisfied first leads to the termination of the algorithm. The termination criteria are described below:

- First, the algorithm cannot exceed the maximum number of GA generations limit which has been set to 50 generations for this work. This limit has been adapted from the standalone hybrid algorithm in Ref. [1–3].

- Second, the algorithm ends if there is no change in the non-dominated design set for 10 consecutive GA generations, adapted from Ref. [1–3]. In this case, the algorithm is unable to find any new non-dominated designs for a few consecutive generations, hinting towards a wastage of computational effort in continuing further with the algorithm.

- Third, if the average distance between consecutive non-dominated design points for 10 consecutive GA generations remains constant, the algorithm ends. This termination criteria signifies that there is no improvement in the spread of the Pareto frontier, which may lead to a wastage of computational effort if continued further with the algorithm. Even if there is any significant improvement after 10 generations, the additional computational cost incurred cannot be justified.

3.2.3 Choosing the Population Size

The multi-fidelity approach involves building surrogate models for all objective and constraint functions. The sample size recommended to train a surrogate model tends to depend on the number of design variables present in the sample data / problem in consideration [33]. Hence, it seems plausible to assume that the GA population size varies proportionately with the number of design variables in the problem. The author recommends that the minimum appropriate number of points in the GA population should be 8 times the number of total design variables (denoted by n), i.e., $8n$. This value for the population size could be treated as an initial approximate number to get a fairly good Pareto frontier with reasonable number of "high-fidelity" function evaluations. For instance, if the problem has 10 design variables (n=10), then the GA population size should be 80 to get a Pareto frontier showing a fairly good spread in the objective space. This technique also takes care of the requirement of the modified two-branch tournament selection technique to set the population size as a multiple of 8 (refer to Section 3.1.2). Increasing the population size beyond the $8n$ formulation could lead to a wider spread in the Pareto frontier but

at the expense of more "high-fidelity" function evaluations and, hence, computational cost.

Section 5.4 conducts experiments with different GA population sizes on the problem of interest – the 'greener' aircraft design problem – to illustrate the reason for settling on a population size that depends on the number of design variables as a multiple of 8, given by the $8n$ guideline.

4. TEST PROBLEMS

This work uses two test problems – versions of three-bar and ten-bar truss problems – to demonstrate the applicability of the multi-fidelity approach to solve simple constrained multi-objective MDNLP problems. The test problems serve as a base to test out the proposed algorithm before applying it to solve a complex "greener" aircraft design problem.

Each test problem is first solved using Kriging as the surrogate modeling strategy. This followed by solving the same test problem using RBF as the alternate surrogate modeling strategy. Kriging models are replaced by RBF models in the latter part of this work due to the higher computational cost involved in solving for the Kriging hyper-parameters. The Kriging models provide good approximation for the objective and constraint values, however, are computationally expensive when compared to RBF models. Because the proposed algorithm requires building a surrogate model for each objective and constraint function, constructing multiple Kriging models every generation sometimes consumes more computational time than the actual "high-fidelity" evaluations for these test problems. This impacts the computational runtime of the multi-fidelity approach, making the computationally cheaper RBF models more suitable for employment in the proposed algorithm.

The following test problems intend to demonstrate the efficacy of the multi-fidelity hybrid algorithm to solve multi-objective optimization problems, while also indicating an increased compatibility of the RBF surrogate modeling strategy with the proposed multi-fidelity approach (as compared to the Kriging surrogate modeling strategy) to solve such problems.

4.1 Three-Bar Truss Problem

The three-bar truss problem solved here is a constrained multi-objective MDNLP problem with six design variables – three continuous and three discrete. The two competing objectives for this problem include minimizing the weight of the truss while simultaneously minimizing the deflection of the free node. The deflection of a node is calculated as the resultant of the sum of all deflections in both the x and y directions. The continuous variables signify the cross-sectional area of the three bars, ranging from 1e-6 cm^2 to 5 cm^2. The discrete variables vary from 1 to 4, signifying the material selection properties of these bars, where the discrete integer values represent Aluminium, Titanium, Steel, and Nickel respectively. This leads to the availability of 4^3 (= 64) combinations of possible material choices. The yield stress for every bar acts as a constraint for the problem (total three constraints), not allowing the stress in the bar to go beyond that upper limit.

Since the three-bar problem has only 64 possible combinations of discrete variables, a search for finding the actual non-dominated designs is conducted by using a gradient-based method (instead of the hybrid approach) for every material choice combination possible for the three bars. The following sub-section investigates the actual Pareto frontier for the three-bar truss problem by using the weighted sum approach for solving multi-objective optimization problems. Ideally, this actual Pareto frontier should coincide with the one obtained using the standalone hybrid algorithm.

4.1.1 Investigating Actual Pareto Frontier

The actual Pareto frontier for the three-bar truss problem is obtained by varying the importance of the two objective functions in a gradient-based method for multi-objective optimization. The weighted sum approach performs a similar task by converting the multiple objectives into a single objective using weights for the objective functions (described in Section 2.2.6).

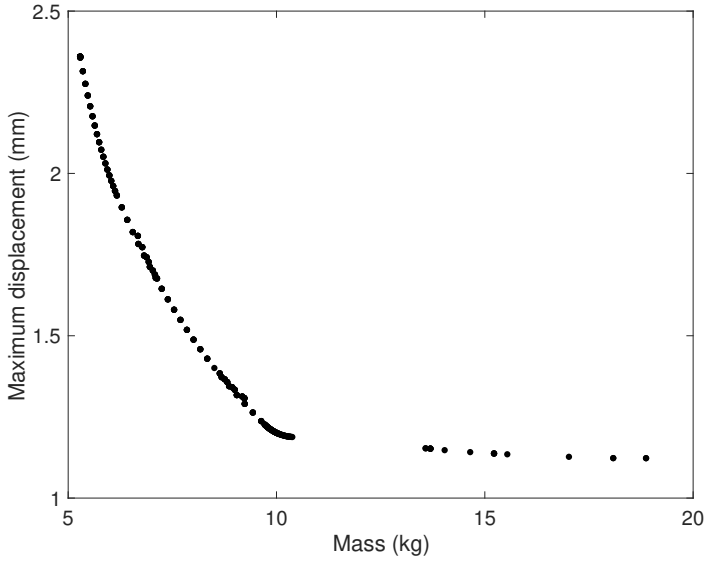

Figure 4.1. Three-bar truss problem: Actual Pareto frontier obtained using the weighted sum approach for each discrete combination.

In this work, the weighted sum approach solves the constrained three-bar truss problem using MATLAB's *fmincon* solver with varying weights for the two objectives given by the formulation – [w_a, 1 - w_a], where w_a = {0.00,0.01,...,0.99,1.00}. The weights for the two objectives are hence given by the vectors – [0.00,1.00], [0.01,0.99],..., [0.5,0.5],...,[0.99,0.01], [1.00,0.00]. Each combination of these weights leads to a single point in the objective space, for a specific combination of discrete variables. The weighted sum approach hence conducts gradient-based search for all 101 weight pairs corresponding to each of the 64 possible discrete combinations possible. Fig. 4.1 shows the Pareto frontier obtained using the gradient-based approach. The non-dominated set consists of 348 designs. Hence, this approach is feasible only for a problem with small number of total possible discrete combinations (64 for this problem), making it infeasible for a problem with larger number of possible discrete combinations. Moreover, the gradient-based approach requires an initial point as input for its local search procedure, leading to different solutions with different initial points.

4.1.2 Solving Three-Bar Truss Problem using Multi-Fidelity Approach

The following two sections solve the three-bar truss problem using the multi-fidelity approach, implementing Kriging surrogate modeling strategy and RBF strategy in sequence. The three-bar truss problem is set up with the GA population size limited to 48 individuals. This is consistent with the approach presented in Section 3.2.3, which recommends the population size to be $8n$, where n is the number of design variables – equivalent to 6 in this case. The upper limit for the number of generations is set to 50. The probability of crossover is set to 0.5 while the mutation rate is fixed to be 0.005. The number of bits chosen for the continuous and discrete variables are 8 and 2 respectively. The three-bar truss problem requires constructing five surrogate models for every GA generation, one for each objective function, and the remaining three corresponding to each problem constraint.

4.1.2.1 Implementing Kriging Models

This section employs Kriging surrogate modeling strategy to construct the surrogate models. The algorithm uses parallel computation to build five Kriging models for each generation in this problem. The resulting Pareto frontier for the three-bar truss problem using multi-fidelity hybrid algorithm with Kriging surrogate models is shown in Fig. 4.2.

The Pareto frontier for this problem shows a large spread across the plot (refer to Fig. 4.2), leading to 235 trade-off designs with a total of 902 "high-fidelity" function evaluations. The analysis of the non-dominated design set shows that an increase in the weight of the three-bar truss system is accompanied by a similar increase in the cross-sectional area of the bars. The material configuration for all three bars gradually shifts to Steel as we move from left to right along the Pareto frontier, with the maximum mass design having a configuration of two Steel bars and one Nickel bar, shown in Fig. 4.2.

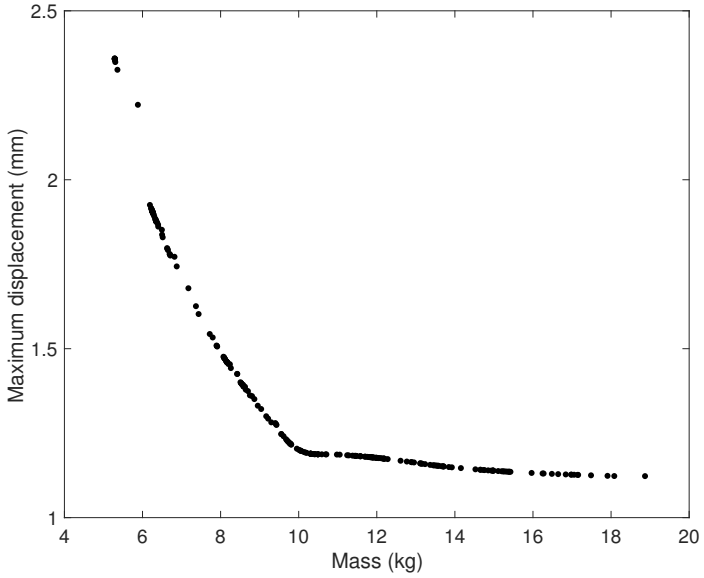

Figure 4.2. Three-bar truss problem: Pareto frontier for multi-fidelity approach using Kriging models.

GA search includes random numbers (not a random search though), which means that two consecutive runs of GA would not find the same results, however, the results would be similar. To assess the repeatability of the multi-fidelity hybrid algorithm, the three-bar problem is run 40 times to see the changes in the spread of the Pareto frontier with every run. It is observed that the spread of the Pareto frontier changes for different runs, which can be attributed to the randomness associated with genetic algorithms (GA) and the different initial population points generated using the LHS sampling strategy. The standalone hybrid algorithm runs for the three-bar problem also show similar behavior with different runs.

The Pareto frontier using the standalone hybrid approach [1, 3] is shown in Fig. 4.3, with 247 trade-off design solutions obtained using 59,147 "high-fidelity" function evaluations. The number of non-dominated design solutions obtained using the multi-fidelity approach is comparable to that obtained by the standalone hybrid approach, with the comparison of their Pareto frontier appearing in Fig. 4.3. For this comparison run, the multi-fidelity approach uses only 1.53% of the total number of

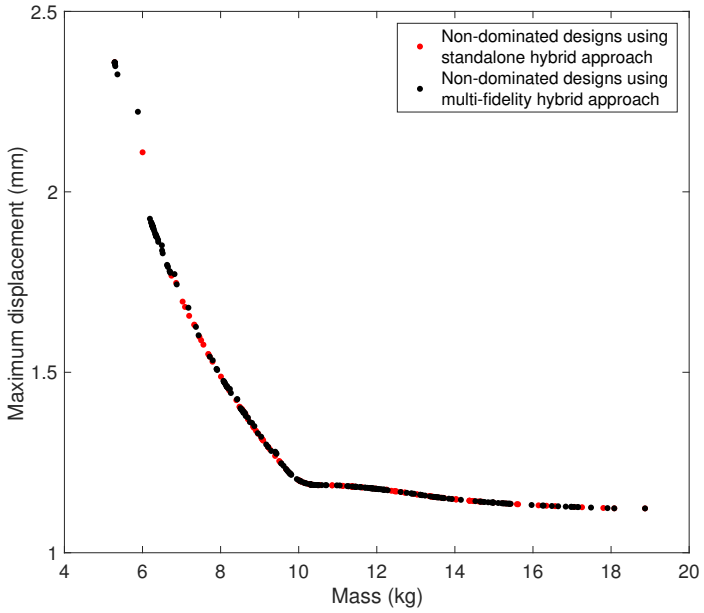

Figure 4.3. Three-bar truss problem: Comparison of Pareto frontiers for the multi-fidelity approach (using Kriging models) and the standalone hybrid approach.

"high-fidelity" analysis performed for the standalone hybrid approach. The number of "high-fidelity" function evaluations for different runs using the Kriging version of the multi-fidelity approach ranges from 443 to 1225 with an average of 884 evaluations, while those for the standalone hybrid approach ranges from 41,567 to 65,416 with an average of 54,418 evaluations.

Fig. 4.4 compares the Pareto frontiers obtained using the multi-fidelity approach (with Kriging models) and the standalone hybrid approach with the actual Pareto frontier obtained using gradient-based approach in Fig. 4.1. Both the multi-fidelity approach and the standalone hybrid approach are able to find non-dominated designs comparable to the ones obtained using the gradient-based approach (implementing weighted sum approach for every possible discrete material combination).

Comparing the computational load of the Kriging version of the multi-fidelity approach with the standalone hybrid approach, the multi-fidelity approach shows an

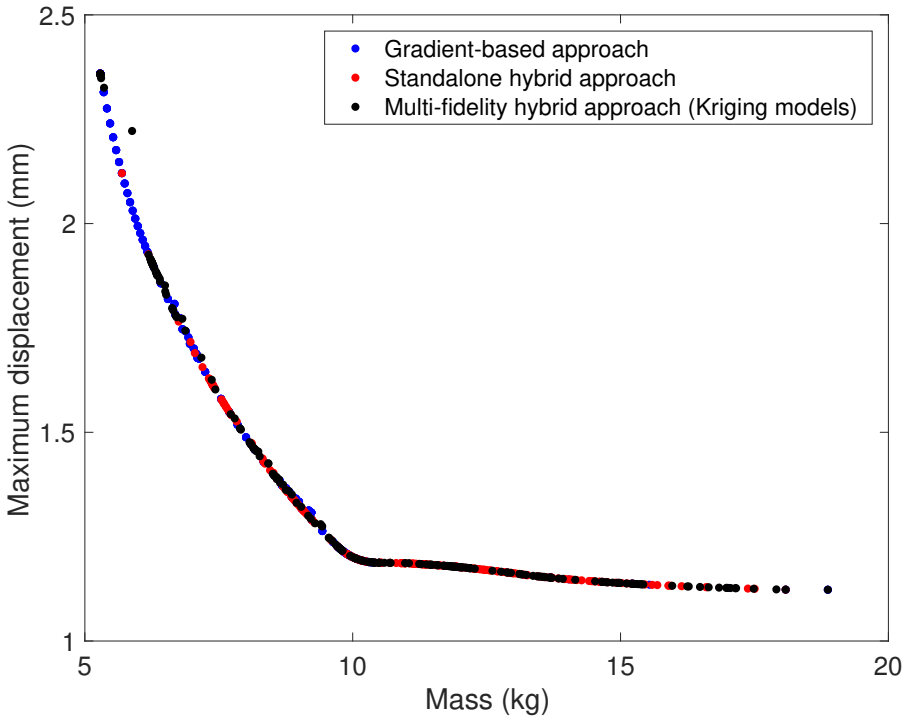

Figure 4.4. Three-bar truss problem: Comparison of the actual Pareto frontier with the ones obtained from the multi-fidelity approach (using Kriging models) and the standalone hybrid approach.

average reduction of 98.38% in the total number of "high-fidelity" function evaluations. However, using Kriging surrogate models, the multi-fidelity approach currently takes an average of 3.8 hours to complete a single run, while the standalone hybrid approach takes only 7.75 seconds on an average to solve the same problem. This anomaly can be attributed to the very "low-fidelity" nature of the three-bar problem, allowing the standalone hybrid approach to perform 'quick' actual function evaluations. On the other hand, the multi-fidelity approach takes time to construct five Kriging models for each GA generation. As the number of sample points usually increase after every GA generation, the computational intensity of the optimization problem to solve for the Kriging hyper-parameters also increases, consuming more computational time to solve the entire constrained, multi-objective MDNLP problem

compared to the standalone hybrid approach. This renders the Kriging version of the multi-fidelity approach impractical even with a lower number of actual "high-fidelity" function evaluations.

4.1.2.2 Implementing Radial Basis Function Models

This section employs RBF surrogate modeling strategy to construct the surrogate models in the multi-fidelity hybrid algorithm. Here, the RBF surrogate models replace Kriging surrogate models due to the computational burden associated with estimating the hyper-parameters for the latter. The algorithm uses parallel computation to build five RBF surrogate models. For this problem, a value of $\sigma = 0.45$ seems to be suitable for building the RBF models, estimated using the trial-and-error approach of comparing the RBF prediction errors to select a suitable σ value (refer to Section 3.1.1). The resulting Pareto frontier for the three-bar truss problem using multi-fidelity hybrid algorithm with RBF surrogate models is shown in Fig. 4.5.

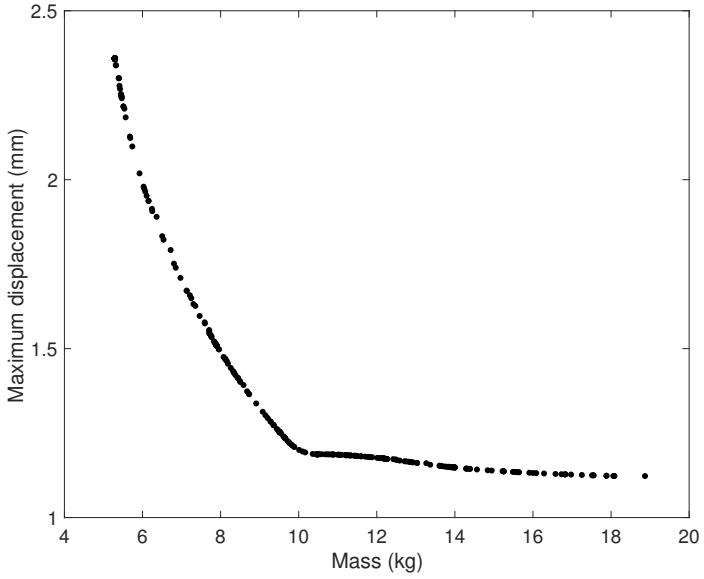

Figure 4.5. Three-bar truss problem: Pareto frontier for multi-fidelity approach using RBF models.

The Pareto frontier for this approach contains 302 non-dominated designs (refer to Fig. 4.5) after making a total of 974 "high-fidelity" function evaluations. The spread of the Pareto frontier obtained using RBF models is comparable to that obtained using the Kriging models (Fig. 4.2). The non-dominated designs show similar trends as the previous ones obtained using Kriging, with the material configuration for all three bars gradually shifting to Steel as we move from left to right along the Pareto frontier. The design with the maximum mass has a configuration of two Steel bars and one Nickel bar.

This RBF version of the multi-fidelity approach is run 40 times (to assess the repeatability of the approach) and the Pareto frontier shows variations in the spread of the non-dominated designs with different runs, a consequence of the nature of the global search algorithm employed – GA, and the generation of different initial population points using the LHS strategy. Significant variations are also observed when the same initial sample points are employed for different runs. The number of "high-fidelity" function evaluations for this approach ranges from 739 to 1,262 with an average of 1,013 evaluations. The RBF version of multi-fidelity hybrid algorithm is able to find more number of non-dominated design solutions when compared to both the standalone hybrid algorithm and the Kriging version of multi-fidelity hybrid algorithm. The Pareto frontier so obtained also shows a slightly wider spread when compared to both these algorithms. Fig. 4.6 shows the comparison between the Pareto front obtained using multi-fidelity approach with RBF and the standalone hybrid approach. Fig. 4.7 compares the Pareto frontiers obtained using the multi-fidelity approach (with RBF models) and the standalone hybrid approach with the actual Pareto frontier obtained using gradient-based approach in Fig. 4.1. As with the Kriging version of the multi-fidelity approach and the standalone hybrid approach, the RBF version is also able to find non-dominated designs comparable to the ones obtained using the gradient-based approach (implementing weighted sum approach for every possible discrete material combination).

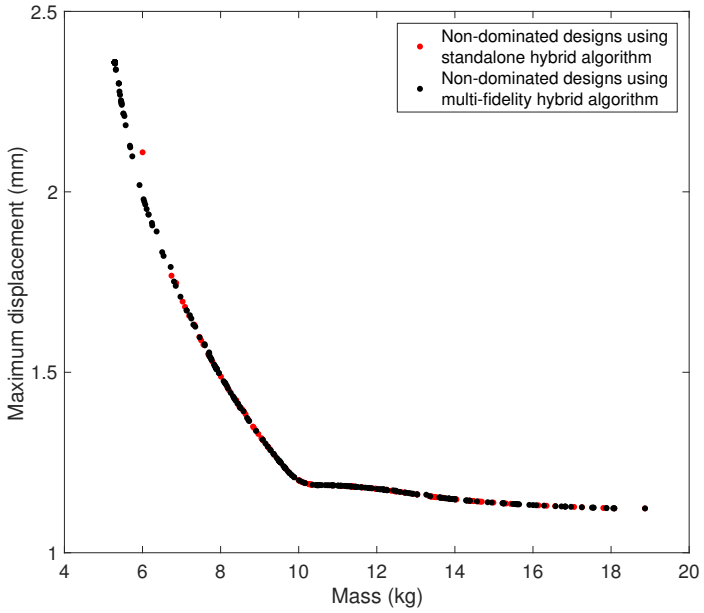

Figure 4.6. Three-bar truss problem: Comparison of Pareto frontiers for the multi-fidelity approach (using RBF models) and the standalone hybrid approach.

Comparing the RBF version of the multi-fidelity approach with the standalone hybrid algorithm, there is a 98.35% reduction in the total number of "high-fidelity" function evaluations for the result shown in Fig. 4.5. The RBF version takes only 12.98 seconds to run (with an average runtime of 12.28 seconds), which is less than the computational time required by the Kriging version of the multi-fidelity hybrid algorithm to solve the same problem. The RBF models employed in this work do not require any optimization problem to be solved to estimate the parameter σ. This leads to quick calculations to model the RBF surrogates, as compared to solving an optimization problem to estimate the hyper-parameters for building Kriging models.

Interestingly, the average runtime of the RBF version of multi-fidelity approach is still more than the average runtime of the standalone hybrid approach, which is 7.75 seconds for the three-bar truss problem. Even with a 98.14% average reduction in the number of "high-fidelity" function evaluations, the multi-fidelity approach requires 1.6

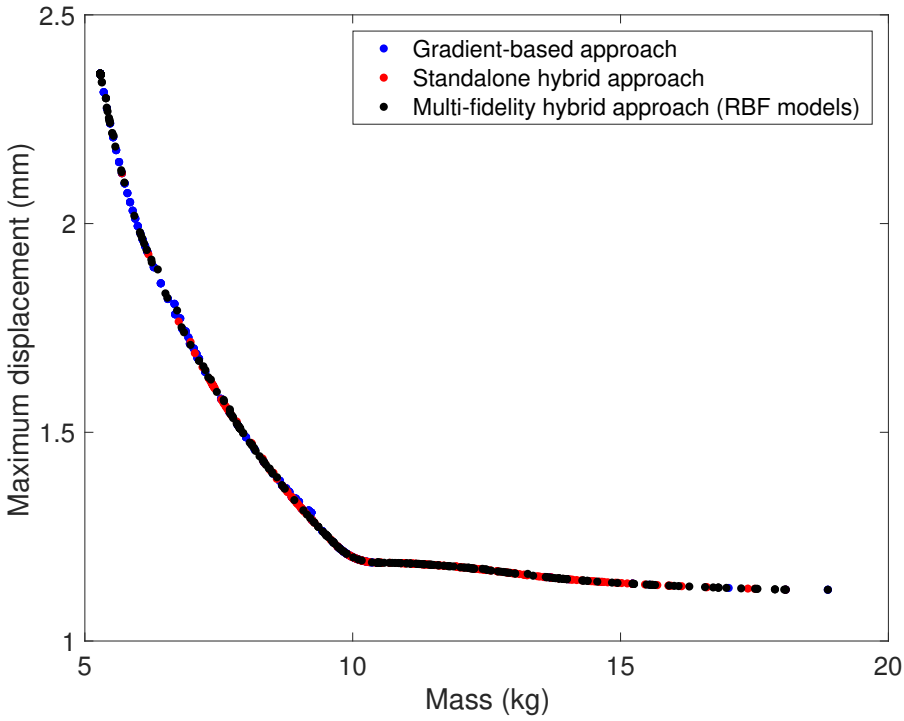

Figure 4.7. Three-bar truss problem: Comparison of the actual Pareto frontier with the ones obtained from the multi-fidelity approach (using RBF models) and the standalone hybrid approach.

times the computational runtime required by the standalone hybrid approach to solve the three-bar truss problem. The extra time consumed by the multi-fidelity approach is due to the formation of five RBF models for every GA generation. The actual function evaluation for the three-bar problem involves simply solving a two-by-two system of linear equations, which are quicker to actually solve than to approximate using RBF models (even though the approximation approach requires far less actual function evaluations). The multi-fidelity approach with RBF modeling provides a higher number of trade-off solutions (with slightly better Pareto frontier) by using an average of only 1.86% of the total number of "high-fidelity" function evaluations required by the standalone hybrid approach (equivalent to a reduction of 98.14%) , while showcasing a slightly increased computational runtime.

Table 4.1. Three-bar truss problem: GA generation-wise RMS error values for objective and constraint RBF models using $\sigma = 0.45$

GA Generations	**RMS Error for RBF Models**				
	Obj. 1	Obj. 2	Constr. 1	Constr. 2	Constr. 3
1	1.1276	0.0036	2.2596	2.4806	2.3990
5	0.6004	0.0024	1.5523	1.7213	1.6319
10	0.4398	0.0018	1.1745	1.2857	1.1149
15	0.4348	0.0017	1.0886	1.1767	2.0885
20	0.4135	0.0016	1.0587	1.1404	2.4043
25	0.4066	0.0016	1.0256	1.1038	3.3280

4.1.2.3 Effect of σ Value on RBF Prediction

This sub-section studies the change in the prediction error of RBF models by using the same σ value throughout the multi-fidelity optimization run, demonstrated using the three-bar truss problem. The prediction error (RMS error value) for each RBF model is calculated using the LOOCV technique for every GA generation with a σ value of 0.45 (as this is the most suitable value found using the trial-and-error approach). Table 4.1 shows the RMS error values for all the objective and constraint RBF models for every 5 generations. This run terminates after 26 GA generations. All the prediction error values (except for the third constraint function) tend to decrease as the GA generations progress, indicating an increase in the quality of function and constraint value predictions. This implies that the RBF models usually tend to better represent the design space as more points are added to the sample set after every GA generation, for the same σ value.

4.2 Ten-Bar Truss Problem

The ten-bar truss problem is actually a scaled-up version of the three-bar truss problem with twenty design variables, ten continuous and ten discrete. The problem minimizes two competing objectives – weight of the ten-bar truss system and resultant displacement of the bars simultaneously. The displacement is taken as the absolute of the maximum calculated displacement among all the elements. The continuous variables signify the cross-sectional diameters of the ten bars, ranging from 0.1 cm^2 to 40 cm^2. The discrete variables signify the material selection properties of these bars, with four choices available for each bar in the problem – including Aluminum, Titanium, Steel, and Nickel. For this problem, there are 4^{10} (=1,048,576) possible material choice combinations compared to only 64 for the three-bar truss problem. The constraint for each bar ensures that the calculated maximum displacement always remains less than the maximum displacement allowed for the bar in the ten-bar system. The ten-bar truss problem requires constructing 12 surrogate models for every GA generation, one for each objective function, and the remaining ten corresponding to each problem constraint.

The GA population has been limited to 160 individuals as per the approach described in Section 3.2.3, with n equivalent to 20. The upper limit for the number of generations is set to 50. The probability of crossover is set to 0.5 while the mutation rate is fixed to be 0.005. The number of bits chosen for the continuous and discrete variables are 8 and 2 respectively.

4.2.1 Implementing Kriging Models

In this section, the Kriging surrogate modeling technique is implemented to solve the ten-bar truss problem using the multi-fidelity hybrid algorithm. This problem requires the generation of twelve Kriging models for every generation. The resulting Pareto frontier for is shown in Fig. 4.8.

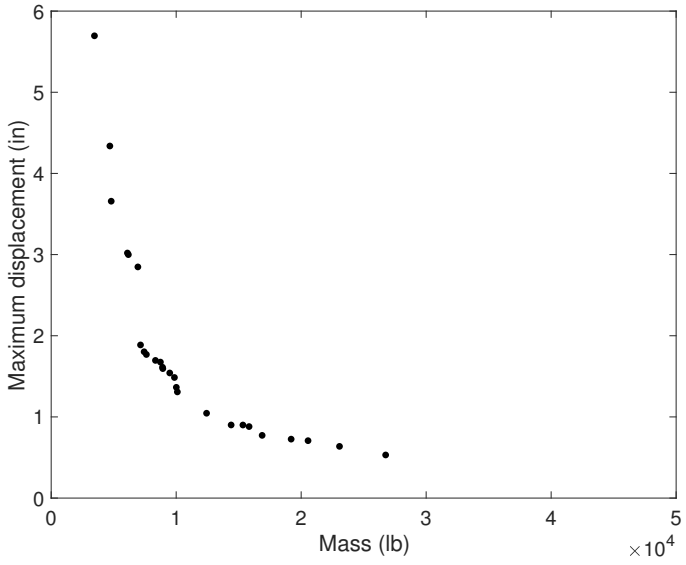

Figure 4.8. Ten-bar truss problem: Pareto frontier for multi-fidelity hybrid approach using Kriging models.

The non-dominated set obtained for the ten-bar truss problem is incomplete, with only 26 trade-off designs in the set. The algorithm took 253.5 hours (approximately 10 days) to complete just 12 GA generations with only 1271 "high-fidelity" function evaluations, forcing the author to terminate the algorithm prematurely. On the other hand, the standalone hybrid algorithm takes an average of 1.71 hours to solve the ten-bar truss problem using an average of 1,466,037 "high-fidelity" function evaluations. This implies that it takes approximately 1.71 hours to perform 1,466,037 "high-fidelity" function evaluations on an average for the ten-bar truss problem, which is in stark contrast to the runtime of 235.5 hours for the multi-fidelity approach with only 1271 similar "high-fidelity" evaluations. Hence, building twelve Kriging models for every GA generation consumes a considerable chunk of the the total computational runtime involved in solving the ten-bar truss problem using the multi-fidelity approach.

4.2.2 Implementing Radial Basis Function Models

This section employs RBF surrogate models to solve the ten-bar truss problem using the multi-fidelity hybrid algorithm. The algorithm uses parallel computation to build the twelve RBF models for this problem. For this problem, a value of $\sigma = 0.5$ is found to be suitable for building the RBF models, based on the trial-and-error approach of comparing the RBF prediction errors to select a suitable σ value (refer to Section 3.1.1). The resulting Pareto frontier appears in Fig. 4.9.

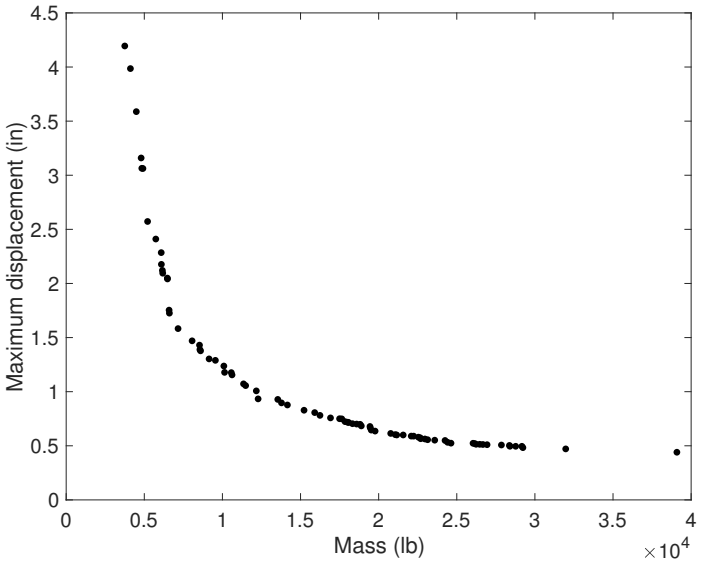

Figure 4.9. Ten-bar truss problem: Pareto frontier for multi-fidelity hybrid approach using RBF models.

With the implementation of the RBF version of multi-fidelity hybrid algorithm, the non-dominated set contains 80 trade-off designs, with 6,537 total "high-fidelity" function evaluations and 9.32 minutes of total runtime, for the result shown in Fig. 4.9. For this approach, the 40 runs conducted to assess repeatability show that the number of "high-fidelity" function evaluations ranges from 3,985 to 8,146 with an average of 6,348 evaluations. The Pareto frontier shows a large spread across the objective space, with each point on the Pareto frontier representing a unique

combination of the ten continuous variables (cross-sectional diameter for each bar) and the ten discrete variables (material choices available for each bar).

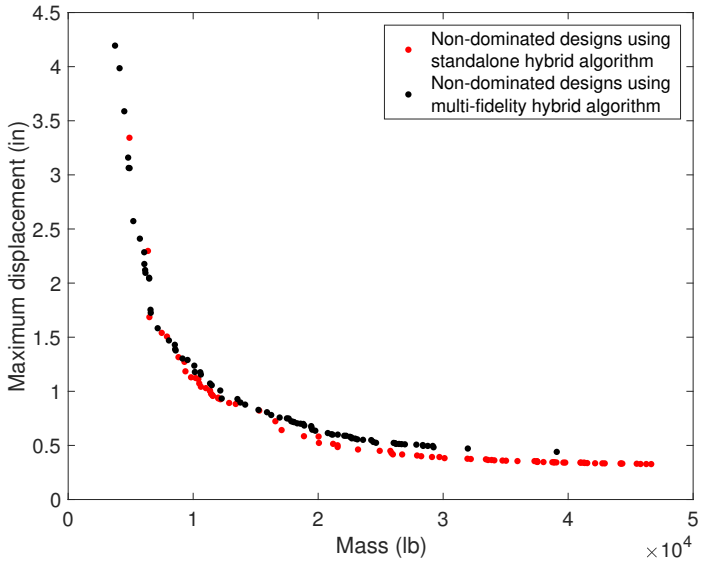

Figure 4.10. Ten-bar truss problem: Comparison of Pareto frontiers for the multi-fidelity approach (using RBF models) and the standalone hybrid approach.

Comparing the results obtained from the multi-fidelity approach with the standalone hybrid approach (refer to Fig. 4.10), the total number of "high-fidelity" function evaluations show an average reduction of 99.57%. This is accompanied by an average reduction of 89.36% in the ten-bar problem runtime – decreasing from an average of 102.43 minutes (standalone hybrid approach) to 10.90 minutes (multi-fidelity approach). The number of trade-off designs in the non-dominated set is similar – 80 for the multi-fidelity approach (with 6,537 "high-fidelity" evaluations and 9.32 minutes of runtime) and 81 for the standalone hybrid approach (with 946,579 "high-fidelity" evaluations and 68.27 minutes of runtime) considering the results presented in Fig. 4.10. However, there is a difference in the spread of the Pareto frontier for the two approaches, visible in Fig. 4.10. The Pareto frontier for RBF version of the multi-fidelity approach shows a reduced spread when compared to the standalone hy-

brid approach. This could be attributed to the fact that the multi-fidelity approach does not perform a "high-fidelity" analysis for every point, essentially giving rise to a trade-off between obtaining a larger spread of the Pareto frontier and the computational cost associated with the problem. However, the multi-fidelity approach does find a few better non-dominated designs in terms of one of the two objectives. The top-left point on the Pareto frontier for the multi-fidelity approach could not be obtained using the standalone hybrid approach for any run. This design point indicates minimum mass for the ten-bar truss system, with a maximum net displacement among all the designs in the problem design space.

When comparing the Kriging version with the RBF version of the multi-fidelity hybrid algorithm, the runtime observed for the latter is approximately 3,280 times less than the runtime for the terminated Kriging version with only 12 completed GA generations. This observation indicates that it is plausible to favor the implementation of RBF models over Kriging models for solving the ten-bar truss problem using the proposed multi-fidelity hybrid algorithm.

4.3 Conclusion

On the basis of the performance of Kriging and RBF versions of the multi-fidelity hybrid algorithm to solve the three-bar and ten-bar truss test problems, the author recommends using the RBF version of the multi-fidelity hybrid algorithm to solve all constrained multi-objective MDNLP problems. A comparison of the performance of the two surrogate modeling approaches to solve the test problems appears in Table 4.2.

For this work, the RBF version of the multi-fidelity hybrid algorithm outperforms the Kriging version by reducing the computational cost incurred to solve for the surrogate hyper-parameters. As discussed in previous sub-sections, the high computational cost associated with the Kriging version of the multi-fidelity hybrid algorithm can be attributed to the need to solve an optimization problem for estimating the Kriging

Table 4.2. Comparison between Kriging and RBF versions of the multi-fidelity hybrid approach for test problems – three-bar truss and ten-bar truss.

Test Problem	**Three-bar Truss**		**Ten-bar Truss**	
Comparison Parameters	**Kriging Version**	**RBF Version**	**Kriging Version (aborted)**	**RBF Version**
Non-dominated Designs	235	302	12	80
"High-fidelity" Evaluations	902	974	923	6,537
Computational Runtime (in hours)	3.8	0.004	253.5	0.16

hyper-parameters. The complexity of the optimization problem increases with an increase in the number of design variables or sample design points or both, making this optimization problem even harder to solve. On the other hand, RBF surrogate modeling does not require solving any optimization problem to estimate σ – the only parameter associated with Gaussian function RBF models. The parameter σ is estimated based on the trial-and-error approach of comparing prediction errors for different σ values (refer to Section 3.1.1), leading to 'quick' calculations for building the RBF surrogate models. Although Kriging models tend to be more accurate in modeling the design space, the RBF models are better suited for this work due to their less computationally intensive nature.

Because the main motivation of this work is to reduce the computational cost (and hence the computational time) associated with solving constrained multi-objective MDNLP problems using a hybrid approach, the Kriging surrogate model based multi-fidelity approach seems infeasible for use in the proposed multi-fidelity hybrid algorithm. The author concludes that the RBF surrogate modeling technique outperforms the Kriging technique for the proposed multi-fidelity approach, and recommends using the RBF technique to build the surrogate models to solve constrained multi-objective MDNLP problems when using the presented multi-fidelity approach.

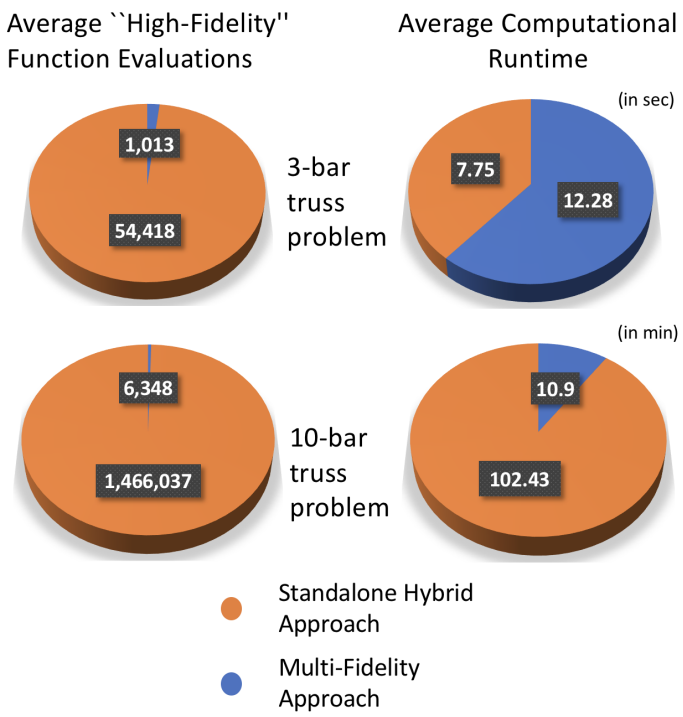

Figure 4.11. Comparison of "high-fidelity" evaluations and computational runtime for the multi-fidelity approach (using RBF models) with the standalone hybrid approach – three-bar and ten-bar truss problems.

Fig. 4.11 uses pie charts to visualize the small fraction of "high-fidelity" function evaluations and computational runtime required by the multi-fidelity approach (with RBF surrogates) compared to the standalone hybrid approach. For the three-bar problem, the multi-fidelity approach uses only 1.86% of "high-fidelity" function evaluations required by the hybrid approach for the three-bar problem. However, as previously discussed, the runtime for the three-bar problem increases for the multi-fidelity approach (in comparison to the runtime of the standalone hybrid approach for the same problem), an attribute of the very "low-fidelity" nature of this problem. For the ten-bar problem, the multi-fidelity approach uses only 0.43% of "high-fidelity" function evaluations, and only 10.64% of the computational runtime (analogous to computational cost) required by the hybrid approach.

Interestingly, for the ten-bar problem (which is a "higher-fidelity" problem when compared to the three-bar problem), the reduction in the computational runtime obtained using the multi-fidelity approach (with RBF surrogates) is not proportional to the reduction in the "high-fidelity" function evaluations performed. This behavior can be attributed to the requirement of building new (or updating existing) RBF models in every GA generation to approximate the objective and constraint function values. Also, as the "high-fidelity" database grows larger, some computational time might be consumed in looking for existing "high-fidelity" information for design points in the database. The computational time required to build the RBF models and to look up the "high-fidelity" database (in some cases) adds up to the computational time required to perform "high-fidelity" function evaluations, leading to a disproportional decrease in the computational runtime compared to the "high-fidelity" analyses conducted for solving constrained multi-objective MDNLP problems using the multi-fidelity approach (with RBF surrogates).

Because the test problem results demonstrate the ability of the multi-fidelity approach (using the RBF surrogate modeling technique) to successfully solve constrained multi-objective MDNLP problems, the author further pursues this approach to solve the 'greener' aircraft design problem.

5. AIRCRAFT DESIGN PROBLEM

The 'greener' aircraft design problem is a step forward to determine which future technologies (including composite structures, laminar flow technologies) will be of utmost importance for designing a fuel-efficient, environment friendly, and economically viable aircraft. Through this problem, the ability of an optimization algorithm to sift through different discrete technology combinations with varying continuous design variables can be tested, making it a plausible constrained MDNLP problem in the aerospace engineering discipline.

Lehner and Crossley [14] investigated the 'greener' aircraft design problem as a constrained MDNLP problem using the two-branch tournament GA (without hybridization), followed by a single-objective hybrid approach [**?**]. Roy [1–3] investigated this problem using a multi-objective hybrid approach. This work re-solves this problem to demonstrate the capability of the multi-fidelity hybrid algorithm to solve a constrained multi-objective MDNLP problem with reduced computational cost. The 'greener' aircraft design problem is the "highest-fidelity" problem solved in this work using the proposed multi-fidelity hybrid algorithm.

Here, the aircraft design problem employs the NASA sizing code FLOPS [79] to perform the sizing and performance calculations of the candidate aircraft designs. The Flight Optimization System (FLOPS) [79] is a multidisciplinary system of computer programs for conceptual and preliminary design and evaluation of advanced aircraft concepts. This software consists of ten primary modules: 1) weights, 2) aerodynamics, 3) engine cycle, 4) analysis, 5) propulsion data scaling and interpolation, 6) mission performance, 7) takeoff and landing, 8) noise footprint, 9) cost analysis, and 10) program control. For this work, FLOPS acts as the "highest-fidelity" application that is used to enable aircraft analysis.

FLOPS accepts a set of both continuous and discrete design variables as input and returns the aircraft performance (fuel burn), environment (NO_X and CO_2 emissions) and economic metrics (total operating cost) as outputs. Simple models simulating the potential 'greener technologies' are modeled using MATLAB and then integrated with FLOPS for their performance calculations. The goal of the aircraft sizing problem is to develop an aircraft with 2940 nmi design range with a seat capacity of 162 seats in two classes, similar to a Boeing 737 or Airbus A320 type aircraft. The following section describes the modeling of the discrete technologies and their integration with FLOPS software to evaluate candidate aircraft designs.

5.1 Simulating Discrete Technologies

The aircraft optimization study here involves the modeling of three discrete technologies. These discrete technologies are modeled on current technology development efforts to reduce the environmental impact of commercial aircraft, like NASA's Subsonic Fixed Wing Project [80]. These discrete technologies tend to serve as promising options for reducing drag, reducing empty weight and improving engine efficiency. The set of potential technologies considered here is the same set considered in Ref. [1–3, 14], except for the engine technologies. Discrete engine technologies such as direct driven turbofan (DDF), geared turbofan (GTF), contra-rotating ducted turbofan (CRTF), and open rotor (OR) engine are not considered in this work due to the difficulties associated with modeling them in FLOPS. However, this work uses a 'baseline' engine to model engine(s) for every candidate aircraft design in the algorithm.

All the three discrete technologies are modeled using MATLAB and can be easily integrated with FLOPS for their performance assessment on different candidate aircraft designs. Table 5.1 lists the discrete technologies considered in this study.

Table 5.1. Discrete technologies.

Laminar Flow Technologies	**Engine position**	**Composite Material Choices**			
		Wing	Fuselage	Nacelle	Tail
NLF-Wing	2 wing	Yes	Yes	Yes	Yes
HLFC-Wing	2 fuselage	No	No	No	No
HLFC-Wing + Nacelle	2 wing + 1 fuselage				
HLFC-Wing + Tail	3 fuselage				
HLFC-Wing + Tail + Nacelle	4 wing				
NLF-Wing + HLFC-Tail	2 wing + 2 fuselage				
NLF-Wing + HLFC-Nacelle	1 fuselage				
NLF-Wing + HLFC-Tail + HLFC-Nacelle	4 wing + 1 fuselage				

5.1.1 Composite Materials

Composite materials reduce the empty weight of an aircraft by making the airframe lighter, leading to a decrease in the aircraft specific fuel consumption. However, the decrease in fuel consumption comes at the cost of increased production and manufacturing cost. The design problem employs four discrete variables using 1 bit each corresponding to the application of composite materials to the aircraft wing, fuselage,

nacelle, and tail. This study employs the weight factors suggested by the FLOPS Reference Manual [81] to account for the use of composite material on different components. Consequently, the increase in component maintenance and operating cost is handled by FLOPS and no multiplication factor needs to be manually included in FLOPS for incorporating these costs.

5.1.2 Number of Engines and their Position

This work acknowledges eight possibilities for the number of engines and their placement on the aircraft, represented by a single discrete variable coded with 3 bits, as shown in Table 5.1. This discrete technology choice is modeled using the default FLOPS commands for engine placement given in the FLOPS Reference Manual [81].

5.1.3 Laminar Flow Technologies

Skin friction drag contributes to approximately 50 percent of the total aerodynamic drag of an aircraft. Maintaining laminar flow and preventing transition to turbulent flow acts as a key factor for reducing the skin friction drag. Natural Laminar Flow (NLF) technology and Hybrid Laminar Flow Control (HLFC) technology work to reduce the skin friction drag by using two different techniques.

Natural laminar flow (NLF) focuses on the shape of the airfoil to delay the transition to turbulent flow by creating a favorable pressure gradient over a long part of the airfoil. This technique does not require any additional equipment to maintain laminar flow over the wing, but, leads to a 5% increase in the manufacturing and maintenance cost of the aircraft due to the required airfoil shape. Natural laminar flow can only be applied to low-swept wings due to cross-flow instabilities [14, 82]. Also, the Tollmien-Schlichting instability [14, 82] causes the laminar flow to be limited to a maximum of 50% of the airfoil chord length. NLF does not add any extra weight penalty to the aircraft.

Table 5.2. Design penalties for laminar flow technologies.

		% Laminar Flow	Manufacturing Cost	Maintenance Cost	Weight
Wing	NLF	Up to 50%	5% increase	5% increase	No change
Wing	HLFC	Up to 50%	50% increase	50% increase	150% increase in air conditioning system
Nacelle	NLF	NA	NA	NA	NA
Nacelle	HLFC	60%	50% increase	50% increase	500 lbs increase in air conditioning system
Tail	NLF	NA	NA	NA	NA
Tail	HLFC	60%	50% increase	50% increase	20% increase in air conditioning system

Hybrid laminar flow control (HLFC) employs a suction technique to maintain laminar flow over the wing. This technique removes the boundary layer air by suction through minute holes on the skin surface. The suction is applied only on the leading edge of the wing to prevent the cross-flow instabilities, making it applicable to swept wings. However, the suction system, which forms the backbone of HLFC, adds additional weight and cost penalties to the aircraft. The weight of the suction system – for the case of HLFC being applied on the wing – is simulated by a introducing a 150% increase in the air-conditioning system weight. This simulation is based on the similar need of both the suction and air-conditioning system to generate a pressure differential to perform their tasks [14,83]. HLFC technique is also applicable to engine nacelle and the aircraft tail, with the increase in weight and cost summarized in Table 5.2 [3, 14] . Hence, hybrid laminar flow control is much more effective in delaying boundary layer transition when compared to natural laminar flow, but this advantage comes at an increased weight, and increased manufacturing and maintenance costs.

A single discrete variable coded with three bits showcases eight different combinations of laminar flow technologies, depending on the type of laminar flow technology

(NLF or HLFC) and its application on different component(s) (wing, nacelle, tail). These combinations appear in Table 5.1. Natural Laminar Flow (NLF) is applied only to the wing of the aircraft. Hybrid Laminar Flow Control (HLFC) is applied to the wing, nacelle, and tail of the aircraft in various combinations. A MATLAB code calculates the percentage of laminar flow on the wing based on the process described in Ref. [14], using the wing sweep and flight conditions information.

5.2 Problem Formulation

The aircraft design problem focuses on the simultaneous minimization of two different pairs of competing objectives – the first pair being total fuel carried and the total operating cost of the aircraft, and the second pair being Nitrogen Oxide (NO_X) emissions and the total operating cost of the aircraft. The total fuel carried is proportional to the fuel burnt by an aircraft during a mission. For every pound of fuel consumed by an aircraft, the engines produce about 3.2 pounds of CO_2. Hence, the total fuel carried is an index of the CO_2 emissions of an aircraft. An airline always tends to reduce the operating cost of an aircraft to increase profit on any mission segment. The total operating cost acts as a problem objective to find potential aircraft designs that could minimize this cost metric (maximizing airline's profit) while keeping a check on the environmental emissions (here, CO_2 and NO_x emissions) of the aircraft.

The problem consists of ten continuous design variables and six discrete design variables, making a total of 16 design variables. The continuous variables include the wing and the engine design variables. The specific upper and lower limits of these continuous design variables (appears in Table 5.3) are based on the work in Ref. [1–3], with each continuous variable coded using 5 bits. This work models the engine technology using incremental values for all the engine continuous variables ($\Delta designvariable$) - except thrust - from a 'baseline' engine design. Table 5.4 lists the 'baseline' engine design parameters along with the function to obtain the final

Table 5.3. Continuous variables.

Design Variables	**Lower Bound**	**Upper Bound**
Aspect Ratio	8	12
Taper Ratio	0.3	0.5
Thickness to Chord Ratio	0.09	0.17
Wing Area [ft^2]	1000	1500
Wing Sweep at 25 % [deg]	0	40
Thrust per Engine [lbs]	20000	30000
By-Pass Ratio (BPR)	0	10
Turbine Inlet Temperature (TIT) [R]	0	500
Overall Pressure Ratio (OPR)	0	20
Fan Pressure Ratio (FPR)	0	0.4

engine design variable value for a candidate aircraft. For instance, if the bypass ratio continuous variable ΔBPR = 5, then the engine bypass ratio value for a particular aircraft design is the sum of the bypass ratio for the 'baseline' engine design and half of the bypass ratio continuous variable, i.e., 'Baseline' + ΔBPR/2 = (10 + 5/2) = 12.5. This incremental type of modeling for the engine variables allows compatibility with all the other GA coded variables. This problem has 1,024 possible discrete technology combinations. Table 5.1 lists the all the possible discrete technologies for this work.

The problem here involves four constraints to meet the aircraft performance requirements as well as FAA requirements. The take-off and landing distances of the candidate aircraft designs are limited to 8,500 feet and 7,000 feet respectively. The landing gear length is limited to 150 inches to ensure that the wing-mounted engines meet the minimum required clearance above the ground. There is also a constraint on the total fuselage fuel capacity of every candidate aircraft, limiting it a maximum of 28,800 pounds.

Table 5.4. Design variables for engine modeling.

Engine Design Variables	**Baseline Engine Design Parameters**	**Engine Design Variable Values** ($\Delta - Cont.\ variable\ value$)
By-Pass Ratio	5	5 + ΔBPR/2
Turbine Inlet Temperature [R]	3010	3010 + ΔTIT
Overall Pressure Ratio	35	35 + ΔOPR
Fan Pressure Ratio	1.6	1.6 + ΔFPR

The multi-fidelity approach requires constructing six RBF models for every GA generation in this design problem, one for each objective function, and the remaining four corresponding to each problem constraint. For this problem, a value of $\sigma = 0.5$ is found to be suitable for building the RBF models, based on the trial-and-error approach of comparing the RBF prediction errors to select a suitable σ value (refer to Section 3.1.1).

In cases where FLOPS cannot "close" a design for the combination of design variables describing the aircraft, the objectives (the fuel burn, NO_X emissions, and total operating cost) are assigned high values of 10^5 to ensure that they are not selected in the future GA generations.

The approach presented in Section 3.2.3 governs the GA population size. According to this approach, a good initial approximation of the GA population size could be given by $8n$, where n is the number of problem design variables. For this problem, sixteen design variables ($n = 16$) lead to a population size of 128 individuals (= 8×16). The upper limit for the number of generations is set to 50 for this problem. The probability of crossover is set to 0.5 while fixing the mutation rate to 0.005. The maximum GA generation limit, crossover probability, and mutation probability are similar to the ones used in the standalone hybrid algorithm in Ref. [1–3]. The max-

imum number of function evaluations for the SQP minimization has been limited to the default value of 100 times the total number of continuous variables.

5.3 Results

5.3.1 Total Fuel Carried vs Total Operating Cost

Fig. 5.1 shows the Pareto frontier for the case of simultaneously minimizing the total fuel carried by an aircraft and the total operating cost, without considering the aircraft NO_X emissions. The multi-fidelity approach finds 44 non-dominated designs by conducting a total of 6,492 "high-fidelity" function evaluations using FLOPS [79]. The total fuel carried by an aircraft is an index of the CO_2 emitted by an aircraft during a mission (as for every pound of fuel consumed, the engines produce about 3.2 pounds of CO_2). The Pareto frontier consists of designs employing combinations of composite structures, eight different engine placement configurations, and a mix of eight different laminar flow technologies, modeled as a part of the 'greener' technology initiative described in the previous section.

The following specific discussion of the non-dominated design results is based upon the technology modeling approach employed in this work. Assuming those are correct, the following observations are made for different non-dominated designs along the Pareto frontier. To reduce the CO_2 emissions, all the non-dominated aircraft designs opt for a two-engine configuration (two wing-mounted engines), along with NLF technology on wings and HLFC technology on the tail and nacelles. Every aircraft in the non-dominated design set opts for a non-composite structure for both the tail and nacelles, while different combinations of composite wings and composite fuselage are visible along the Pareto frontier. The use of composite structures leads to a decrease in the fuel consumption (due to the assumptions of reduced empty weight) and an increase in the total operating cost (due to the assumptions of increased manufacturing and maintenance costs associated with composites). As we move from

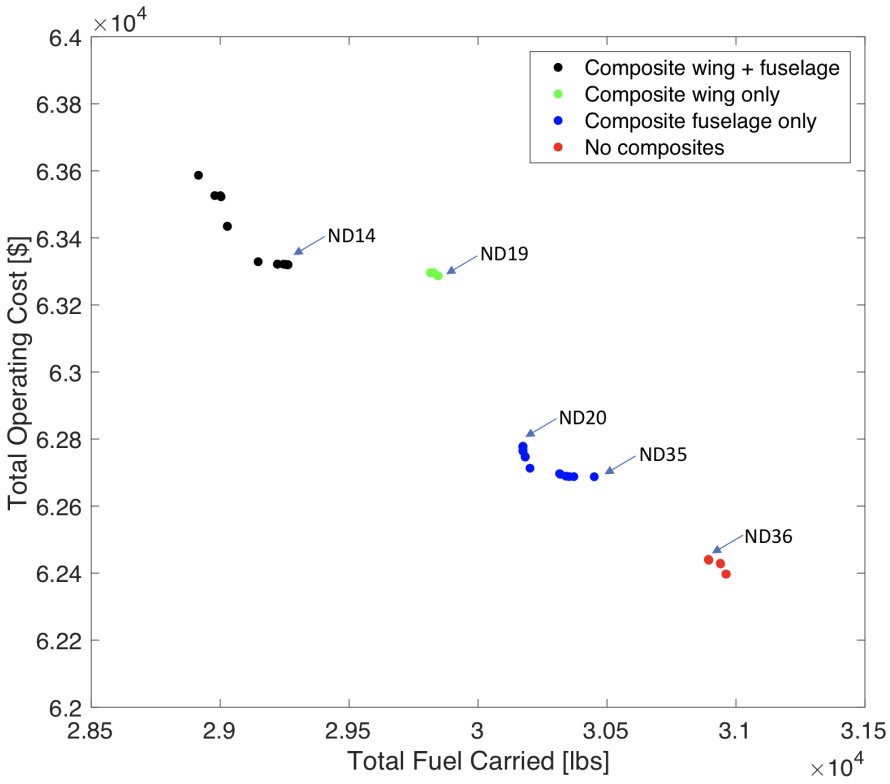

Figure 5.1. Pareto frontier for the objective pair – total fuel carried (index for CO_2 emissions) and total operating cost.

left to right along the Pareto frontier (refer to Fig. 5.1), designs tend to opt for an all non-composite structure configuration to reduce the total operating cost.

The design with minimum CO_2 emissions (represented by the leftmost point in Fig. 5.1) selects a two-engine configuration (wing-mounted engines) along with composite wings and fuselage. Consequently, it has the maximum total operating cost due to the increased manufacturing and maintenance costs associated with the all-composite wing and fuselage structure. Analyzing the engine design variables, it features the highest TIT and FPR among all designs, along with one of the lowest OPR – all contributing to the minimum CO_2 characteristic of this design. Also, the NLF technology on wing and HLFC technology on tail and nacelles work to maintain laminar flow on the aircraft wings, tail and nacelles – reducing the drag induced by

skin friction which ultimately decreases the aircraft fuel consumption (analogous to reduction in CO_2 emissions).

The design with the minimum total operating cost (represented by the rightmost point in Fig. 5.1) selects a two-engine configuration (wing-mounted engines) without any composite material components. Even though this design has the lowest total operating cost amidst all non-dominated designs, it still opts for NLF technology on wings along with HLFC technology on the tail and nacelles. This signifies the importance of employing laminar flow technologies in future 'greener' aircraft designs, based upon the technology modeling employed here.

Examining the region near the fourteenth to nineteenth non-dominated designs (represented by ND14 and ND19 respectively in Fig. 5.1), it is visible that the design points seem to align themselves along the horizontal axis, indicating that a considerable reduction in total fuel carried by an aircraft is possible for a nominal increase in its total operating cost. Considering designs ND14 and ND19, a 0.05% increase in total operating cost leads to a 2% reduction in the aircraft CO_2 emissions as we move from right to left along the Pareto frontier. This is attributed to a change in the composite material application, changing from a composite wing and fuselage configuration (ND14) to a composite wing and non-composite fuselage configuration (ND19). The non-composite material nature of the fuselage in ND19 reduces the manufacturing and maintenance costs for the aircraft – reducing the total operating cost; while increasing the aircraft weight (leads to an increase in the fuel burn) – raising the CO_2 emissions.

Similarly, examining the bottom right portion of the Pareto frontier, a substantial reduction in total fuel carried is achievable with a minimal increase in total operating cost. Analyzing designs ND35 and ND36 (refer to Fig. 5.1), a 1.45% reduction in total fuel carried (CO_2 emissions) is possible with a 0.4% increase in total operating cost. This reduction in total fuel carried is equivalent to a 442.9 lb decrease in the total CO_2 emissions per aircraft per trip. This is attributed to a shift from a no-composite configuration (ND36) to a composite fuselage configuration (ND35), reducing the

aircraft weight and hence, the fuel burn (CO_2 emissions). This is accompanied by a decrease in the wing area (directly reducing aircraft weight) and the thrust per engine (directly decreasing the fuel burn). Moreover, comparing ND35 and ND44 (rightmost point in the Pareto frontier) can lead to a further decrease of 69.2 lbs in CO_2 emissions per trip (a total of 512.1 lbs per trip) with an additional increase of just 0.07% in the total operating cost (a total of 0.47%).

An interesting region from an airline's point of view would be the near the points ND19 and ND20, where a nearly vertical portion is visible in the Pareto frontier (refer to Fig. 5.1). Moving from left to right in this region, a substantial decrease in total operating cost is possible for a marginal increase in the total fuel carried (CO_2 emissions) by the aircraft. A change from a composite wing configuration (ND19) to a composite fuselage configuration (ND20) is responsible for this decrease in total operating cost. This is accompanied by an increase in the wing sweep and thrust per engine from ND19 to ND20.

The aircraft geometry design variables show a general trend as we move from left to right along the Pareto frontier (refer to Fig. 5.1). The wing area and thrust per engine tends to increase for a specific combination of composite structures along the Pareto frontier. The engine BPR and OPR generally increases from the costliest design to the cheapest design, along with a decrease in the FPR. Also, none of the designs employ HLFC technology for all components together – wings, tail and nacelles. This can be attributed to the weight and cost penalties associated with HLFC (especially for aircraft wings), which suggest that the designs employing NLF technology for wings are more affordable as compared to those employing HLFC.

The aircraft design problem for this objective is run 40 times to see the changes in the spread of the Pareto frontier with every run. The initial population points generated using the LHS strategy tends to influence the spread of the Pareto frontier for different runs. Also, the randomness associated with the two-branch tournament GA adds to the changes observed in the Pareto frontier spread. This means that even

with same set of initial population points, an exact same set of design solutions is not obtained for different runs of the algorithm.

5.3.1.1 Comparison with Standalone Hybrid Approach

This section compares the results obtained using the multi-fidelity approach (using RBF models) with the ones obtained using the standalone hybrid approach for the total fuel carried (index of CO_2 emissions) vs total operating cost objective pair to solve the 'greener' aircraft design problem.

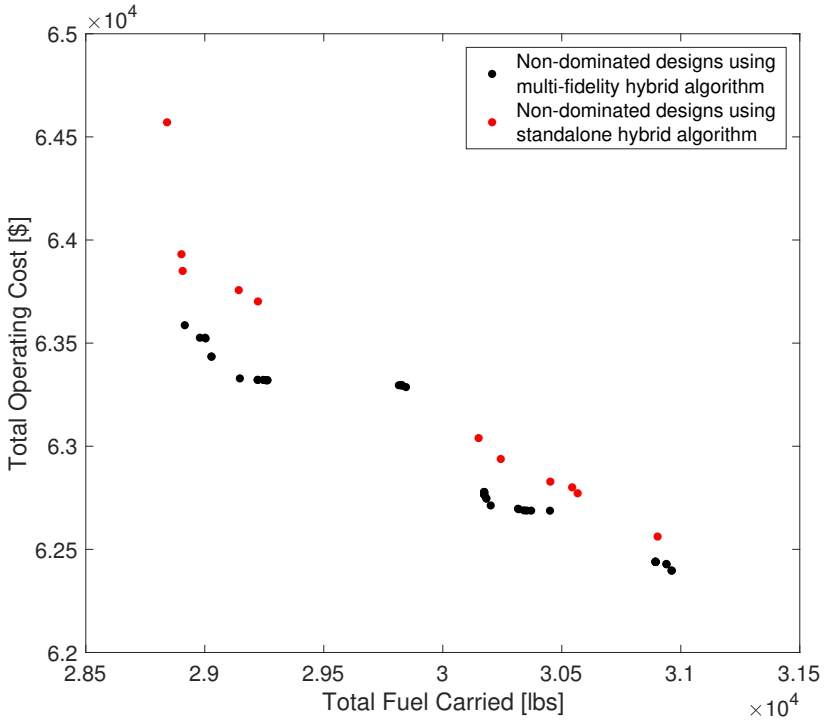

Figure 5.2. Comparison of Pareto frontiers obtained using multi-fidelity hybrid approach and standalone hybrid approach for the objective pair – total fuel carried (index for CO_2 emissions) and total operating cost.

Fig. 5.2 collectively plots the resulting Pareto frontiers from both the approaches. The standalone hybrid approach finds 11 non-dominated designs using a total of

1,739,004 “high-fidelity” function evaluations (with an average of 1,711,620 “high-fidelity” evaluations over all 10 runs). The total algorithm runtime for obtaining this solution set is 578.10 minutes (with an average runtime of 571.22 minutes over all 10 runs). On the other hand, the multi-fidelity approach finds 44 non-dominated design solutions using 6,492 “high-fidelity” function evaluations (6,490 “high-fidelity” evaluations on average over all 40 runs), with a total runtime of 4.62 minutes (5.76 minutes on average over all 40 runs), for the results presented in Fig. 5.1 and Fig. 5.2. This approach shows an average reduction of 99.62% in the total number of “high-fidelity” function evaluations along with a 98.99% average reduction in the algorithm runtime – leading to a proportional decrease in the computational cost associated with solving the ‘greener’ aircraft design problem (fuel carried vs total operating cost objective pair) presented in this work. Fig. 5.3 illustrates the decrease in the number of “high-fidelity” evaluations and computational runtime for the multi-fidelity approach – with respect to the standalone hybrid approach.

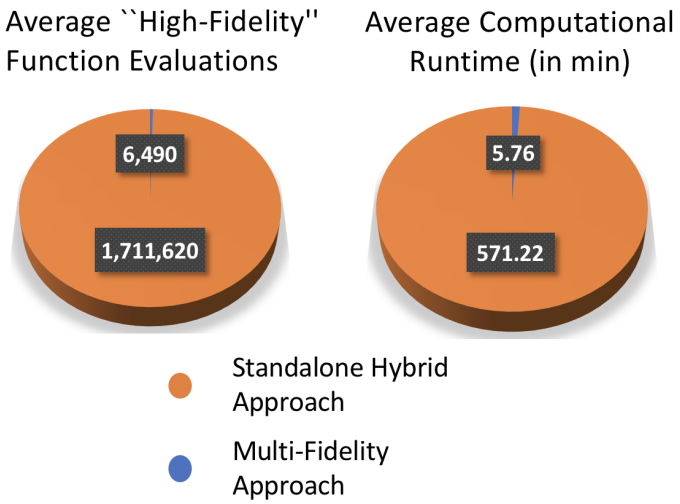

Figure 5.3. Comparison of “high-fidelity” evaluations and computational runtime for the multi-fidelity approach with the standalone hybrid approach – total fuel carried (index for CO_2 emissions) and total operating cost.

Hence, the multi-fidelity approach (using RBF models) is able to find more non-dominated designs with a less number of "high-fidelity" analyses and lesser runtime. This behavior appears to be an attribute of the discontinuities present in the FLOPS sizing tool. The RBF models tend to smoothen FLOPS' internal discontinuities by building a continuous approximation model, allowing the multi-fidelity hybrid algorithm to perform local search on a continuous design space. However, the standalone hybrid algorithm encounters discontinuities while performing local search by directly employing the FLOPS sizing tool for design evaluations, causing the standalone hybrid algorithm to penalize some potentially good designs by stopping the local search because of one of these discontinuities. This leads to a smaller set of non-dominated design solutions for the standalone hybrid approach when compared to the multi-fidelity approach.

Analyzing the design solutions in Fig. 5.2, the non-dominated designs obtained using the standalone hybrid approach constitute a Pareto frontier with a wider spread as compared to the ones obtained using the multi-fidelity approach. However, the multi-fidelity approach is able to find a number of trade-off designs with low operating cost and high CO_2 emissions (visible as black points in the bottom right portion of Fig. 5.2). The standalone hybrid approach is unable to find these designs even after multiple runs. This behavior is also likely due to the presence of discontinuities in the FLOPS sizing tool, causing the standalone hybrid algorithm to penalize some potentially good designs in the local search step.

An interesting point to note in Fig. 5.2 is that the trade-off design solutions obtained from the multi-fidelity approach (represented by black points) are visibly closer to both the horizontal and vertical axes of the figure, when compared to those obtained using the standalone hybrid approach (represented by red points). This indicates that the black points are better design solutions as they essentially dominate the solutions represented by the red points. Hence, the design solutions from the multi-fidelity approach dominate the ones obtained using the standalone hybrid approach. This behavior is also an attribute of the presence of discontinuities in the

FLOPS sizing tool. These discontinuities hinder the standalone hybrid algorithm from performing local search in certain regions of the design space. On the other hand, the continuous design space approximation generated by the RBF models allows the multi-fidelity algorithm to search a discontinuity-free design space, allowing the multi-fidelity approach to find a better set of non-dominated design solutions (obtainable with the multi-fidelity hybrid algorithm). Subsequently, the CO_2 emission value for the lowest CO_2 emission designs obtained using both the approaches (represented by leftmost black and red points) are comparable. However, the multi-fidelity approach is able to find this design for a lower total operating cost than that found by the standalone hybrid approach. This reduces the spread of the Pareto frontier for the multi-fidelity approach, as seen in Fig. 5.2.

5.3.2 NO_X Emissions vs Total Operating Cost

Fig. 5.4 shows the Pareto frontier for the case of simultaneously minimizing the NOx emissions and the total operating cost incurred by the aircraft, without considering the fuel consumption. The multi-fidelity approach finds 88 non-dominated designs by conducting a total of 6,100 "high-fidelity" function evaluations using FLOPS [79]. The non-dominated designs show the best combination of aircraft geometry design variable values and different discrete 'greener' aircraft technologies to reduce the aircraft NO_X emissions.

The following specific discussion of the non-dominated design results is based upon the technology modeling approach employed in this work. Assuming those are correct, the following observations are made for different non-dominated designs along the Pareto frontier. All the non-dominated aircraft designs employ NLF technology on wings along with HLFC technology on the tail and nacelles. These aircraft opt for a non-composite structure for both the tail and nacelles, and a composite structure for fuselage for every aircraft – except one. This exception could be attributed to the mutation operator in the two-branch tournament GA.

The design with minimum NO_X emissions (represented by the leftmost point in Fig. 5.4) selects a three-engine configuration with one fuselage-mounted and two wing-mounted engines, along with composite wings and fuselage. This design features the maximum wing area among all the non-dominated designs. Consequently, it has the maximum total operating cost due to the increased manufacturing and maintenance costs associated with the largest composite wing and the all-composite fuselage structure. The cost associated with operating and maintaining three engines also adds up to the total cost, making this design the costliest to operate. This is an interesting outcome as perhaps no designer would choose three engines for an aircraft to meet the specified mission requirements, however, the impact on the aircraft NO_X emissions is clearly visible when a three-engine configuration is selected while solving the problem using the multi-fidelity hybrid approach.

The design with maximum NO_X emissions (represented by the rightmost point in Fig. 5.4) employs two wing-mounted engines along with a composite fuselage structure, leading to minimum total operating cost amidst all designs. This design also has the lowest thrust per engine among all the designs, leading to very high NO_X emissions. Analyzing the engine design variables, it features one of the highest BPR (second to maximum), TIT (third to maximum), and OPR (maximum), along with the lowest FPR – all contributing to the high NO_X emissions for this design.

Examining the right portion of the Pareto frontier, the design points tend to align themselves in an almost horizontal line, indicating that a substantial decrease in the NO_X emissions can be obtained for a marginal increase in the total operating cost as we move towards the left part of the Pareto frontier. Analyzing the sixty-ninth and eighty-eighth non-dominated designs (represented by ND69 and ND88 respectively in Fig. 5.4), it is observed that a marginal increase of 0.18% in the total operating cost leads to a 71.9% reduction in the aircraft NO_X emissions (from 539.61 lbs to 308.86 lbs per trip). This behavior is an attribute of the higher thrust per engine and very low TIT and OPR values of ND69 (both almost equivalent to their respective 'baseline' engine parameters) when compared to ND88. Higher turbine temperatures lead to

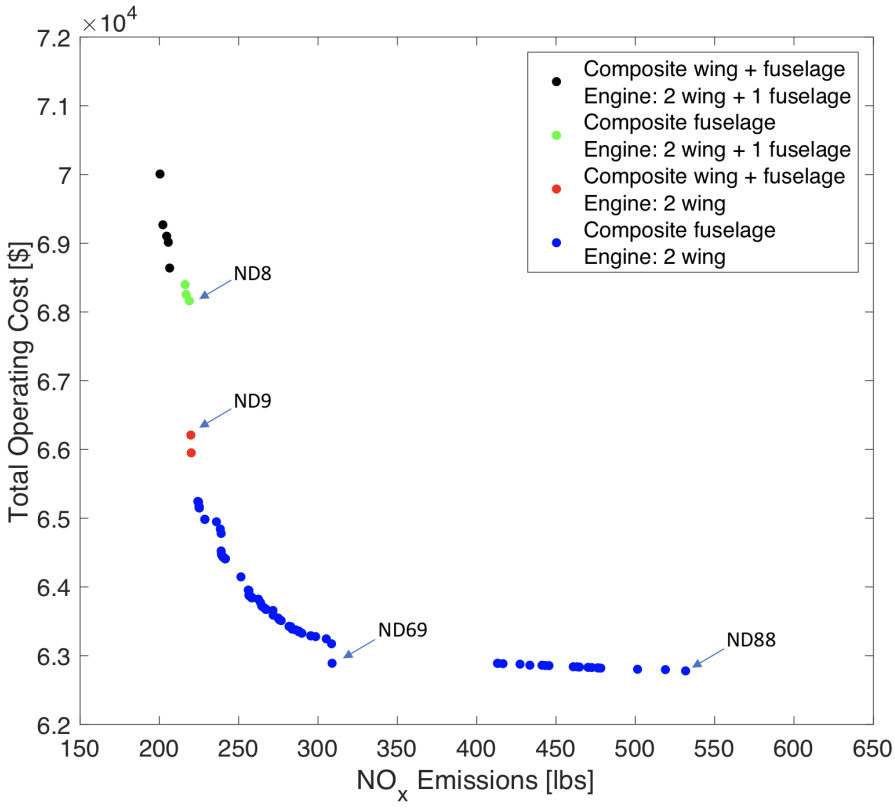

Figure 5.4. Pareto frontier for the objective pair – amount of NO_X emissions and total operating cost.

increased NO_X formation. The TIT of ND88 and ND70 is almost 314 Rankine and 257 Rankine more than that of ND69, leading to the huge difference in NO_X emissions visible in the right part of the Pareto frontier. Additionally, there is a reduction in the wing area along with a simultaneous increase in the aspect ratio and wing sweep from designs ND69 to ND88.

An interesting region from an airline's point of view would be the near the points ND8 and ND9, where a nearly vertical portion is visible in the Pareto frontier (refer to Fig. 5.4). Moving from top to bottom in this region (from ND8 to ND9), a 2.95% decrease in total operating cost is possible for a marginal increase of 0.45% in the NO_X emissions of the aircraft. This behavior is attributed to a shift in the number of engines on the aircraft – from a three-engine configuration (two wing-mounted and

one fuselage-mounted) in ND8 to a two-engine configuration (two wing-mounted) in ND9.

The engine design variables show a general trend as we move from left to right along the Pareto frontier (refer to Fig. 5.4). The BPR and OPR tend to gradually increase – with OPR being maximum at the rightmost end of the Pareto frontier, while FPR gradually decreases as we move from the left end to the right end. For a specific combination of composite structures and number of engines with their position, the thrust per engine tends to decrease as the NO_X emissions of an aircraft increases. The employment of NLF technology on all aircraft wings and HLFC technology on all aircraft tail and nacelles signifies the importance of the laminar flow technologies in 'greener' aircraft design configurations. The highest NO_X emitting design shows an increase of 165.3% emissions as compared to the lowest NO_X emitting design across the Pareto frontier, with an overall decrease in total operating cost of about 11.5%.

For this objective pair, the problem is run 40 times to see the changes in the spread of the Pareto frontier with every run. The spread of the Pareto frontier changes for different runs, which is an attribute of the different set of initial population points generated using the LHS technique and probably because the GA uses some random numbers during the operators.

5.3.2.1 Comparison with Standalone Hybrid Approach

This section compares the NO_X vs total operating cost objective pair results obtained using the multi-fidelity approach (using RBF models) with the ones obtained using the standalone hybrid approach for the 'greener' aircraft design problem. Fig. 5.5 plots the resulting Pareto frontiers from both the approaches together. The standalone hybrid approach finds 28 non-dominated designs using a total of 1,579,997 "high-fidelity" function evaluations (with an average of 1,556,142 "high-fidelity" evaluations over all 10 runs). The total algorithm runtime for obtaining this solution set is 931.67 minutes (with an average runtime of 923.50 minutes over all 10 runs). On

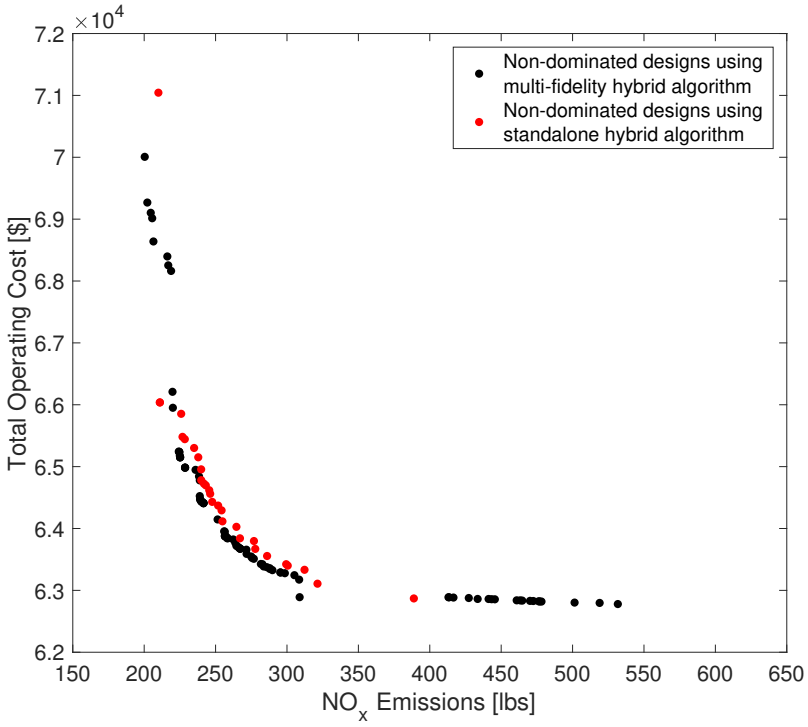

Figure 5.5. Comparison of Pareto frontiers obtained using multi-fidelity hybrid approach and standalone hybrid approach for the objective pair – amount of NO_X emissions and total operating cost.

the other hand, the multi-fidelity approach finds 88 non-dominated design solutions using 6,100 "high-fidelity" function evaluations (5,921 "high-fidelity" evaluations on average from 40 runs), with a total runtime of only 6.25 minutes (7.36 minutes on average from 40 runs). This approach shows an average reduction of 99.62% in the number of "high-fidelity" function evaluations along with a 99.20% average reduction in the algorithm runtime, leading to a proportional decrease in the computational cost associated with solving the 'greener' aircraft design problem presented in this work. Fig. 5.6 illustrates the decrease in the number of "high-fidelity" evaluations and computational runtime for the multi-fidelity approach – with respect to the standalone hybrid approach.

Similar to the observation for the previous objective pair, the multi-fidelity approach (using RBF models) is able to find more non-dominated designs along with a reduction in the "high-fidelity" analyses and computational runtime when compared to the standalone hybrid approach. This behavior is an attribute of the discontinuities present in the FLOPS sizing tool, with the RBF models providing a continuous design space to the multi-fidelity approach for the local search step.

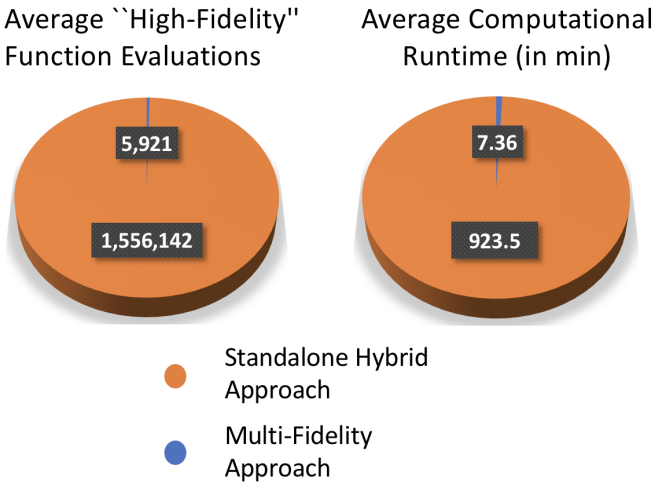

Figure 5.6. Comparison of "high-fidelity" evaluations and computational runtime for the multi-fidelity approach with the standalone hybrid approach – amount of NO_X emissions and total operating cost.

Taking a closer look at Fig. 5.5, the non-dominated designs obtained from the multi-fidelity approach run constitute a much wider Pareto frontier when compared to the one obtained using the standalone hybrid approach. Focusing on the rightmost portion of the Pareto frontier, the multi-fidelity approach finds a number of trade-off designs with high NO_X emissions and low total operating cost (represented by black points in the bottom right portion). The standalone hybrid approach is unable to find these designs even after multiple runs. As was the case in the previous discussion, this behavior is most likely due to the presence of discontinuities in the FLOPS sizing tool, causing the standalone hybrid algorithm to penalize some potentially good designs in the local search step by stopping the search as a result of one (or more) of these

discontinuities. However, the standalone hybrid approach finds a design solution with maximum total operating cost among all the designs in the Pareto frontier (including those from multi-fidelity approach), represented by a red point in the top left corner in Fig. 5.5. The multi-fidelity approach is unable to find this point without compromising the low total operating cost designs (those in the bottom right portion). It seems that the multi-fidelity approach tends to favor the total operating cost objective more than the NO_X emissions objective.

As with the total fuel carried vs total operating cost objective pair, the trade-off design solutions obtained from the multi-fidelity approach (represented by black points) dominate the design solutions obtained from the standalone hybrid approach (represented by red points). This behavior is due to the presence of discontinuities in the FLOPS sizing tool, with the RBF models (employed in the multi-fidelity approach) building continuous approximation models to eliminate their effect on the 'greener' aircraft design problem solution.

5.4 Spread of Pareto Frontier vs Computational Cost

This section signifies the trade-off that exists between obtaining a Pareto frontier with a wide spread across the objective space and the computational cost (essentially the number of "high-fidelity" analyses) associated with the algorithm.

Table 5.5. Comparison of results obtained using different population sizes for the objective pair – total fuel carried and total operating cost.

Comparison Parameters	**Population Size**				
	64 (4n)	**96 (6n)**	**128 (8n)**	**160 (10n)**	**192 (12n)**
Non-dominated Designs	32	34	44	63	97
"High-fidelity" Evaluations	2,827	4,831	6,492	8,123	9,758
Computational Runtime (in sec)	145.94	196.67	277.42	388.64	470.07

Table 5.6. Comparison of results obtained using different population sizes for the objective pair – aircraft NO_X emissions and total operating cost.

Comparison Parameters	**Population Size**				
	64 (4n)	**96 (6n)**	**128 (8n)**	**160 (10n)**	**192 (12n)**
Non-dominated Designs	46	69	88	95	119
"High-fidelity" Evaluations	2,821	3,496	6,100	8,084	9,747
Computational Runtime (in sec)	144.05	179.88	375.33	612.81	722.42

The spread of a Pareto frontier across the objective space is visualized by the maximum (or minimum) value of each objective in a two-objective optimization problem. These maximum (or minimum) values of both the objectives signify the ability of an algorithm to find design solutions on the extreme ends of the Pareto frontier – designs that show most improvement in one objective and least improvement in the other objective – widening the spread of the Pareto frontier. However, to find these extreme trade-off design solutions using the multi-fidelity approach, an increase in the GA population size is required. Hence, increasing the population size leads to an increase in the spread of the Pareto frontier, but at the expense of more "high-fidelity" function evaluations and computational runtime (and computational cost). Table 5.5 and 5.6 compares the non-dominated designs, "high-fidelity" function evaluations, and computational runtime, associated with different GA population sizes for total fuel carried vs total operating cost (first objective pair) and NO_X emissions vs total operating cost (second objective pair) objective pairs respectively. The GA population sizes chosen for comparison are all dependent on the number of problem design variables (n), with the population size incrementing from 64 (analogous to $4n$ for the aircraft design problem) to 192 (analogous to $12n$ for the aircraft design problem).

Fig. 5.7 to 5.10 compare the Pareto frontier obtained for both the objective pairs using different populations sizes ($4n$, $6n$, $10n$, $12n$) with the one obtained using $8n$ population size (based on the approach in Section 3.2.3). Using a population size of 64 points, the Pareto frontiers obtained for both the objectives show a minimal spread compared to the ones obtained using 128 points. In fact, these design solutions (denoted by black points) seem to be dominated by the solutions obtained using 128 points (denoted by red points), appearing in Fig. 5.7. This case requires the minimum number of "high-fidelity" evaluations among all the cases (based on population size), leading to minimum computational runtime. Similarly, for a population size of 96 points (Fig. 5.8), the solutions seem to be dominated by the one obtained using 128 points, with the Pareto frontier having greater spread than the case with 64 points (that too only for the second objective pair), but still being less wider than the one obtained using 128 points. For both these cases, the results look somewhat diminished and unacceptable for the number of "high-fidelity" function evaluations conducted.

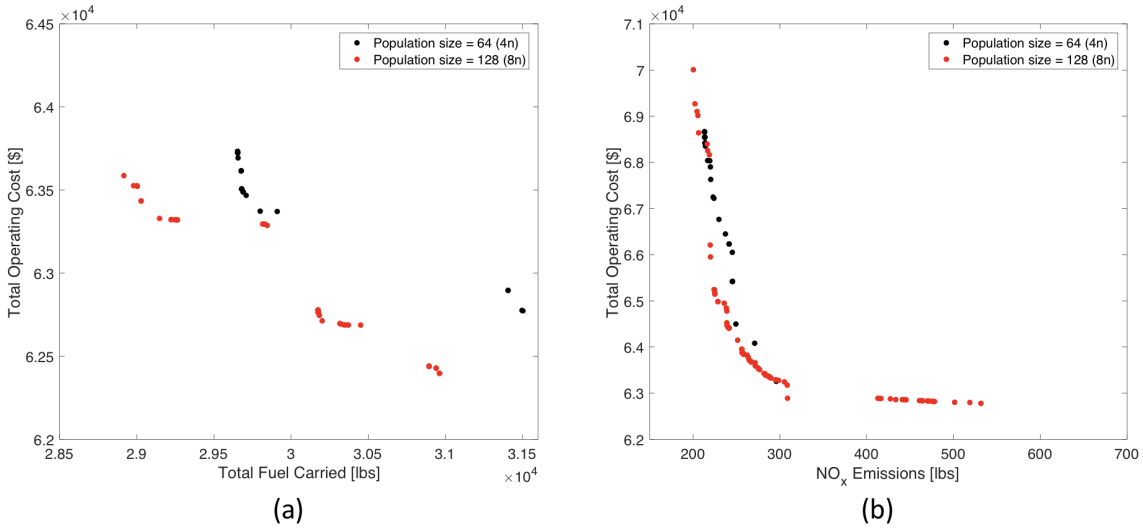

Figure 5.7. Comparison of Pareto frontiers for population size 128 and 64 – (a) Total fuel carried vs total operating cost, (b) NO_X emissions vs total operating cost.

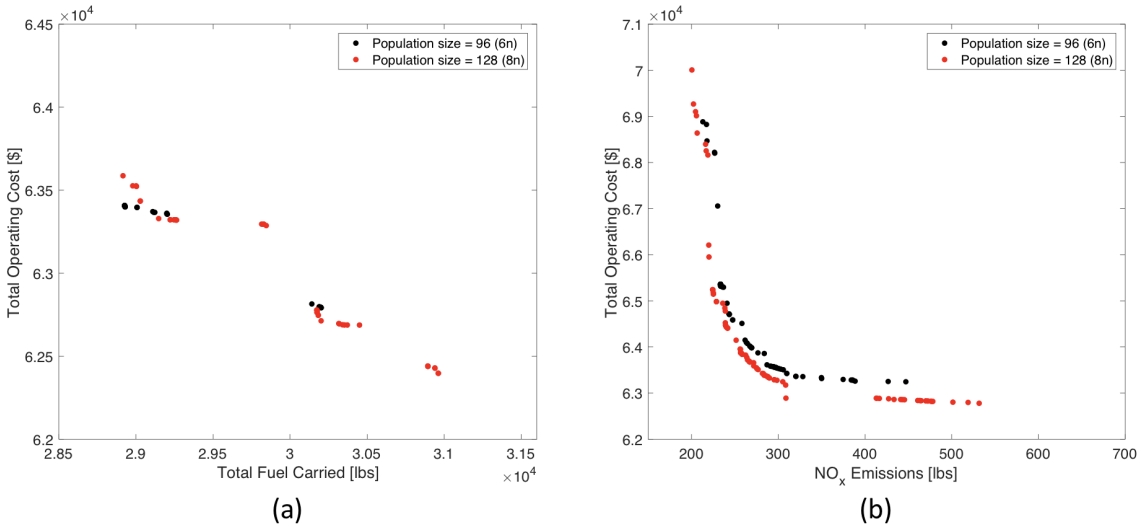

Figure 5.8. Comparison of Pareto frontiers for population size 128 and 96 – (a) Total fuel carried vs total operating cost, (b) NO_X emissions vs total operating cost.

In contrast, the cases with a population size of 160 points (Fig. 5.9) and 192 points (Fig. 5.10) show a wider spread in their Pareto frontiers as compared to the one obtained for 128 points. However, this is accompanied with a respective increase of 40% and 69.4% in the computational time for the first objective pair, and an increase of 63.3% and 92.5% in the computational time for the second objective pair. Examining the Pareto frontier for the first objective pair obtained using a population size of 160 points, the Pareto frontier contains more trade-off designs favoring one of the objectives (visible in the top-left portion of Fig. 5.9a). However, the designs favoring the other objective still seem to be dominated by the solutions obtained using 128 points (visible in the bottom-right portion of Fig. 5.9a) – indicating only minor improvements in the Pareto frontier for an almost 40% increase in cost. Similarly, for the second objective pair with a population size of 160 points, the Pareto frontier shows a wider spread, but with a huge gap in the designs obtained in the bottom-right portion (denoted by black points) of Fig. 5.9b. Examining the Pareto frontiers

obtained using a population size of 192 points, a wider spread is visible for both the objective pairs (Fig. 5.10). However, this is accompanied by a 69.4% and 92.5% increase in computational runtime for first and second objective pair respectively, making this case very expensive when compared to all the other cases.

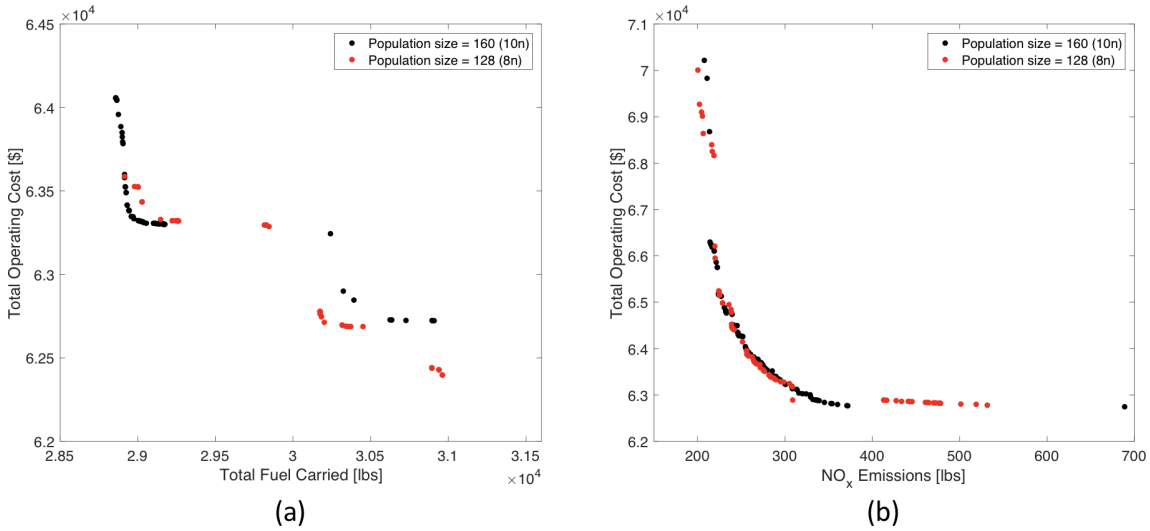

Figure 5.9. Comparison of Pareto frontiers for population size 128 and 160 – (a) Total fuel carried vs total operating cost, (b) NO_X emissions vs total operating cost.

Subsequently, Fig. 5.11 depicts the increment in number of "high-fidelity" function evaluations and computational runtime for different population sizes for both the problem objective pairs. An almost linear increase in the number of "high-fidelity" function evaluations is visible for both objective pairs for a similar increase in population size (refer to Fig. 5.11a). A rather steep gradient is visible in Fig. 5.11b for the computational runtime from 96 ($6n$) points to 160 points ($10n$) for the second objective pair (NO_X emissions vs total operating cost), indicating a disproportional increase in the computational cost incurred for every $2n$ increments in the population size. For the "high-fidelity" aircraft design problem presented in this work, the increase in computational time seems negligible (or minor) for both the objectives,

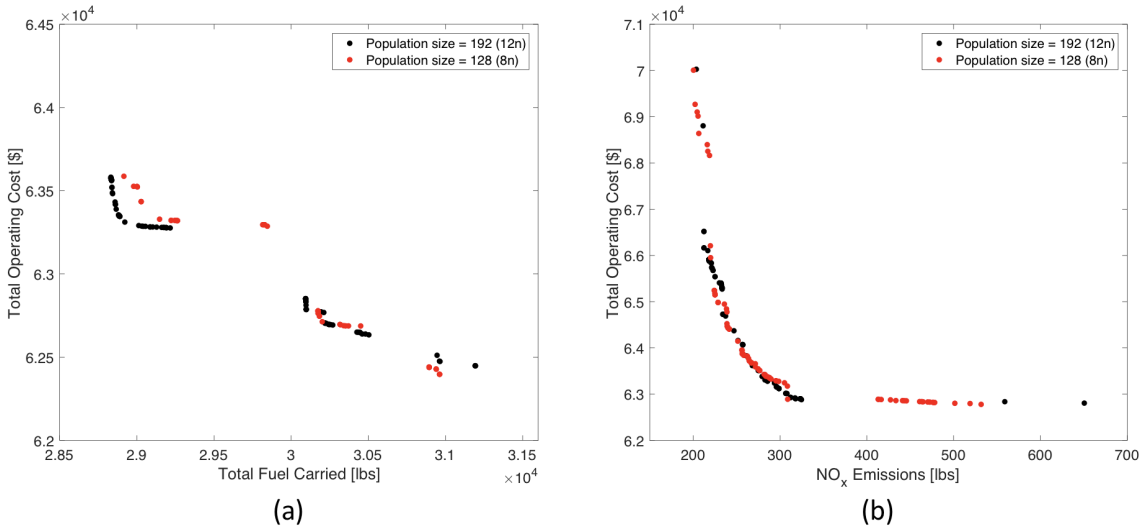

Figure 5.10. Comparison of Pareto frontiers for population size 128 and 192 – (a) Total fuel carried vs total operating cost, (b) NO_X emissions vs total operating cost.

but for a "higher-fidelity" optimization problem, this minor increase in computational runtime – analogous to an increase in the population size – will correspond to a large portion of the computational cost involved in solving the "higher-fidelity" problem.

The above discussion suggests that a trade-off exists between the spread of the Pareto frontier attained for a problem and the computational cost associated with the same. An increase in spread of the Pareto frontier comes at the cost of increased "high-fidelity" function evaluations and increased computational cost. To obtain a Pareto frontier with decent spread using reasonable number of "high-fidelity" function evaluations (with a feasible computational runtime), choosing a GA population size of $8n$ for the multi-fidelity hybrid algorithm seems plausible.

Hence, the author recommends using the $8n$ guideline for choosing the population size for the multi-fidelity approach, where n is the number of problem design variables. This recommendation is based on the comparison of computational runtime incurred for $8n$ population size runs with all other runs, along with spread of the Pareto frontier observed for the $8n$ population size when compared to the ones obtained for

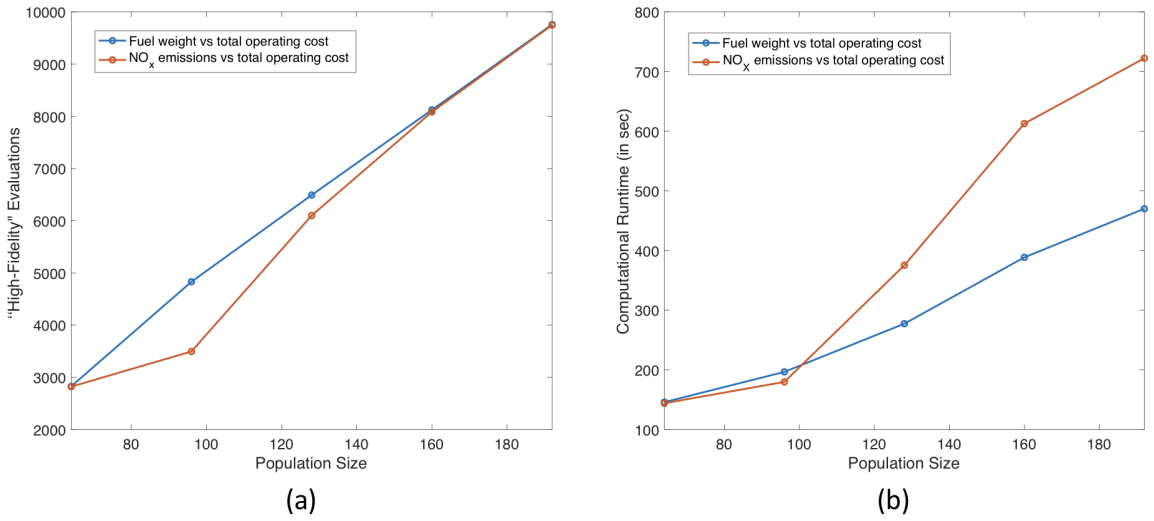

Figure 5.11. Aircraft design problem: (a) "High-fidelity" function evaluations using different population sizes; (b) Computational runtime using different population sizes.

other population sizes. The $8n$ population size also tends to satisfy the condition required for the implementation of the modified two-branch tournament GA – which requires the population size to be a multiple of 8 for conducting multi-objective GA tournament selection and crossover operations.

6. CONCLUSION AND RECOMMENDATIONS

This work presents a multi-fidelity approach to solve constrained multi-objective MDNLP problems by utilizing surrogate models to assist SQP in conducting gradient based local search – coupled with the global search conducted by GA. The developed algorithm works to reduce the computational expense involved with the standalone hybrid algorithm – reducing the number of "high-fidelity" function evaluations required to evolve to a Pareto frontier for a multi-objective problem.

The first version of this algorithm employed Kriging surrogate models to provide "low-fidelity" objective and constraint approximations. Although the Kriging models seemed to provide good approximations for the objective and constraint values in the test problems – the computational burden associated with generating these Kriging models (solving for the Kriging hyper-parameters) rendered this approach impractical for the multi-fidelity hybrid approach. Subsequently, the second version (selected version) employs RBF models to approximate the objective and constraint values ("low-fidelity" approximations). The RBF version of the multi-fidelity hybrid algorithm reduces the computational runtime associated with the standalone hybrid algorithm by at least 89%, leading to a similar decrease in the computational cost. The reduction in the number of "high-fidelity" function evaluations also shows a similar trend with reduction of at least 98%. These reductions vary from problem to problem.

The aircraft design problem demonstrates the efficacy of the multi-fidelity hybrid algorithm to solve complex engineering problems with a reduced computational budget. The algorithm is able to sift through a vast combination of continuous variables (representing aircraft geometry and engine variables) and discrete technologies (composite material application, engine placement, and laminar flow technology) to find economical aircraft designs with reduced fuel burn (CO_2 emissions) and NO_X emis-

sions. For both the objective pairs, all designs opt for natural laminar flow technology on the wings and hybrid laminar flow technologies on the tail and nacelles, indicating the importance of laminar flow technology in enhancing aircraft performance. Although discrete technologies modeled in this work are not very "high-fidelity" in nature, the design solutions provide a good assessment of the discrete technologies that need to be included in the development of a 'greener' economical aircraft.

The non-dominated solution set for the multi-fidelity approach seems to be dependent on the GA population size and the initial GA population generated using the LHS technique. Due to lack of a well-defined criterion for choosing the population size for a multi-objective hybrid approach, the author recommends using the $8n$ approach suggested in this work to set the population size for a problem, where n corresponds to the number of design variables in the problem.

This work also tends to signify a visible trade-off between the number of "high-fidelity" function evaluations (hence computational cost) conducted for obtaining the non-dominated design solutions and the diversity of the Pareto frontier (suggesting its spread in the objective space). Increasing the GA population size seems to widen the spread of the Pareto frontier at the cost of an increase in the number of "high-fidelity" function evaluations (leading to a proportional increase in the computational cost associated with solving the problem).

6.1 Recommendations for Further Research

Although this work demonstrates the ability of the multi-fidelity approach to solve constrained multi-objective MDNLP problems with a limited computational budget, there is a need to conduct a comparison concerning the spread (and quality) of the Pareto frontier and the computational cost of the presented algorithm with other multi-fidelity multi-objective algorithms in existence. Further research would include developing a "higher-fidelity" aircraft design problem to enable a better representation of the impact of the discrete technologies on aircraft performance, probably

modeling the whole problem without using the FLOPS sizing tool. Future work would include adding more discrete technology options to the already existing mix of discrete technologies. The "higher-fidelity" aircraft design problem will create a good platform to further demonstrate the ability of the presented multi-fidelity approach to solve highly complex engineering design problems, while maintaining a low-cost profile.

REFERENCES

REFERENCES

[1] S. Roy and W. A. Crossley. Hybrid multi-objective combinatorial optimization technique with improved compatibility between GA and gradient-based local search. In *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSM*, Indianapolis, IN, 2012. AIAA.

[2] S. Roy and W. A. Crossley. Hybrid approach for multi-objective combinatorial optimization in search of greener aircraft. In *11th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*, Virginia Beach, VA, 2011. AIAA.

[3] S. Roy. Multi-objective optimization using a hybrid approach for constrained mixed discrete nonlinear problems - applied to the search of greener aircraft. Master's thesis, West Lafayette, IN, USA, 2012.

[4] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: NSGA-II. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, PPSN VI, pages 849–858, London, UK, UK, 2000. Springer-Verlag.

[5] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: improving the strength pareto evolutionary algorithm. Technical report, 2001.

[6] A. Kumar, D. Sharma, and K. Deb. *A hybrid multi-objective optimization procedure using PCX based NSGA-II and sequential quadratic programming*, pages 3011–3018. Sept 2007.

[7] H. Satoru, H. Tomoyuki, and M. Mitsunori. Hybrid optimization using direct, ga, and sqp for global exploration. In *2007 IEEE Congress on Evolutionary Computation*, pages 1709–1716, 2007.

[8] X. Hu, Z. Huang, and Z. Wang. *Hybridization of the multi-objective evolutionary algorithms and the gradient-based algorithms*, volume 2, pages 870–877 Vol.2. Dec 2003.

[9] L. Xiang and D. Gang. *BSTBGA: A hybrid genetic algorithm for constrained multi-objective optimization problems*, volume 40, pages 282 – 302. 2013.

[10] X. Zhng, B. Zhang, C. Zhang, and A. Ma. *A multi-objective hybrid genetic algorithm for detecting communities in complex networks*, page 691695. 2016.

[11] A. E. I. Brownlee and J. A. Wright. Constrained, mixed-integer and multi-objective optimisation of building designs by NSGA-II with fitness approximation. *Applied Soft Computing*, 33:114 – 126, 2015.

[12] K. Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001.

[13] K. Deb. Current trends in evolutionary multi-objective optimization. *International Journal for Simulation and Multidisciplinary Design Optimization*, 1(1):1–8, 2007.

[14] S. Lehner and W. A. Crossley. Combinatorial optimization to include greener technologies in a short-to-medium range commercial aircraft. In *ICAS Congress*, 2008.

[15] W. A. Crossley, A. M. Cook, and D. W. Fanjoy. Using the two-branch tournament genetic algorithm for multiobjective design. *AIAA Journal*, 37(2):261, 267, 1999.

[16] S. Lehner. Hybrid optimization for constrained mixed discrete nonlinear problems: An application to the design of an environmental conscious transonic aircraft. Master's thesis, West Lafayette, IN, USA, 2010.

[17] G. N. Vanderplaats. *Multidiscipline Design Optimization*. VR&D, 2007.

[18] K. Schittkowski. NLQPL: A FORTRAN-subroutine solving constrained nonlinear programming problems. *Annals of Operations Research*, 5:485–500, 1985.

[19] M. J. D. Powell. A fast algorithm for nonlinearly constrained optimization calculations. *University of Cambridge, England*, 1977.

[20] V. Pareto. *General Notion of Economic Equilibrium,"Manuale di Economia Politica", Societa Editrice Libraria (translated to English by A.S. Schwier, Manual of Political Economy, Macmillan, New York, 1971)*. Milan, Italy, 1906.

[21] J. S. Arora. Chapter 17 - multi-objective optimum design concepts and methods. In J. S. Arora, editor, *Introduction to Optimum Design (Third Edition)*, pages 657 – 679. Academic Press, Boston, third edition edition, 2012.

[22] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[23] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[24] E. A. Williams and W. A. Crossley. Empirically-derived population size and mutation rate guidelines for a genetic algorithm with uniform crossover. In *Soft Computing in Engineering Design and Manufacturing*, pages 163–172, London, 1998. Springer London.

[25] J. D. Schaffer. *Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms (Artificial Intelligence, Optimization, Adaptation, Pattern Recognition)*. PhD thesis, Nashville, TN, USA, 1984.

[26] C. A. Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, 1:269–308, 1998.

[27] C. A. Coello. An updated survey of evolutionary multiobjective optimization techniques: State of the art and future trends. In *Proceedings of the Congress on Evolutionary Computation*, pages 3–13. IEEE Press, 1999.

[28] C. A. Coello. An updated survey of ga-based multiobjective optimization techniques. *ACM Comput. Surv.*, 32(2):109–143, June 2000.

[29] A. Konak, D. W. Coit, and A. E. Smith. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety*, 91(9):992 – 1007, 2006. Special Issue - Genetic Algorithms and Reliability.

[30] 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, Jun 1994.

[31] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[32] J. P. C. Kleijnen. A comment on blanning's metamodel for sensitivity analysis: The regression metamodel in simulation. *Interfaces*, 5(3):21–23, 1975.

[33] A. I. J. Forrester, A. Sbester, and A. J. Keane. *Engineering Design via Surrogate Modelling*. John Wiley & Sons, Ltd, West Sussex, UK, 2008.

[34] G. G. Wang and S. Shan. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129, 2006.

[35] R. R. Barton and M. Meckesheimer. Chapter 18: Metamodel-Based Simulation Optimization. In *Simulation*, volume 13 of *Handbooks in Operations Research and Management Science*, pages 535 – 574. Elsevier, 2006.

[36] D. G. Krige. A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy*, 52(6):119–139, 1951.

[37] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–423, 1989.

[38] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

[39] M. J. Sasena, P. Papalambros, and P. Goovaerts. Exploration of metamodeling sampling criteria for constrained global optimization. 34:263–278, 01 2003.

[40] M. J. Sasena. Flexibility and efficiency enhancements for constrained global design optimization with kriging approximations, 2002.

[41] S. E. Gano, J. E. Renaud, J. D. Martin, and T. W. Simpson. Update strategies for kriging models used in variable fidelity optimization. *Structural and Multidisciplinary Optimization*, 32(4):287–298, Oct 2006.

[42] M. Li. Robust optimization and sensitivity analysis with multi-objective genetic algorithms: Single- and multi-disciplinary applications, 2007.

[43] D. Huang. Experimental planning and sequential kriging optimization using variable fidelity data, 2005.

[44] D. Huang, T. T. Allen, W. I. Notz, and R. A. Miller. Sequential kriging optimization using multiple-fidelity evaluations. *Structural and Multidisciplinary Optimization*, 32(5):369–382, 2006.

[45] T. Simpson, T. M. Mauery, J. J. Korte, and F. Mistree. Kriging models for global approximation in simulation-based multidisciplinary design optimization. 39:2233–2241, 12 2001.

[46] S. Jeong, M. Murayama, and K. Yamamoto. Efficient optimization design method using kriging model. 42:413–420, 09 2005.

[47] N. A. C. Cressie. *Statistics for Spatial Data*. John Wiley & Sons, New York, 1993.

[48] H. Song and D. A. Lamb. A study on improving the accuracy of kriging models by using correlation model/mean structure selection and penalized log-likelihood function. In *10th World Congress on Structural and Multidisciplinary Optimization*, Orlando, FL, 2013.

[49] S. N. Lophaven, H. B. Nielsen, and J. Sndergaard. DACE - A MATLAB Kriging Toolbox - Version 2.0, 2002.

[50] R. L. Hardy. Multiquadric equations of topography and other irregular surfaces. *Journal of Geophysical Research*, 76(8):1905–1915, 1971.

[51] M. F. Hussain, R. R. Barton, and S. B. Joshi. Metamodeling: Radial basis functions, versus polynomials. *European Journal of Operational Research*, 138(1):142 – 154, 2002.

[52] T. W. Simpson, J. D. Poplinski, P. N. Koch, and J. K. Allen. Metamodels for computer-based engineering design: Survey and recommendations. *Engineering with Computers*, 17(2):129 – 150, Jul 2001.

[53] A. A. Mullur and A. Messac. Extended radial basis functions: More flexible and effective metamodeling. 43:1306–1315, 06 2005.

[54] H. Fang, M. Rais-Rohani, and M. Horstemeyer. Multiobjective crashworthiness optimization with radial basis functions. pages 1306–1315, 06 2005.

[55] H. M. Gutmann. A radial basis function method for global optimization. *Journal of Global Optimization*, 19(3):201–227, Mar 2001.

[56] R. G. Regis and C. A. Shoemaker. A stochastic radial basis function method for the global optimization of expensive functions. *INFORMS Journal on Computing*, 19(4):497–509, 2007.

[57] R. G. Regis and C. A. Shoemaker. Parallel stochastic global optimization using radial basis functions. *INFORMS Journal on Computing*, 21(3):411–426, 2009.

[58] S. M. Wild and C. A. Shoemaker. Global convergence of radial basis function trust-region algorithms for derivative-free optimization. *SIAM Review*, 55(2):349–371, 2013.

[59] J. Müller, C. A. Shoemaker, and R. Pich. SO-MI: a surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems. *Computers and Operations Research*, 40(5):1383 – 1400, 2013.

[60] J. Müller. MISO: mixed-integer surrogate optimization framework. *Optimization and Engineering*, 17(1):177–203, Mar 2016.

[61] V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New York, 1998.

[62] N. J. Kolencherry. Multi-fidelity optimization strategies using genetic algorithms and sequential kriging surrogates. Master's thesis, West Lafayette, IN, USA, 2011.

[63] MATLAB. The mathworks inc. matlab version 2017a, 2017.

[64] M.E. Johnson, L.M. Moore, and D. Ylvisaker. Minimax and maximin distance designs. *Journal of Statistical Planning and Inference*, 26(2):131 – 148, 1990.

[65] A. J. Booker, J. E. Dennis, P. D. Frank, D. B. Serafini, V. Torczon, and M. W. Trosset. A rigorous framework for optimization of expensive functions by surrogates", journal="structural optimization. 17(1):1–13, Feb 1999.

[66] D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21(4):345–383, Dec 2001.

[67] K. Holmström. An adaptive radial basis algorithm (arbf) for expensive black-box global optimization. *Journal of Global Optimization*, 41(3):447–464, Jul 2008.

[68] E. Davis and M. Ierapetritou. A kriging based method for the solution of mixed-integer nonlinear programs containing black-box functions. *Journal of Global Optimization*, 43(2):191–205, Mar 2009.

[69] J. Müller and J. D. Woodbury. GOSAC: global optimization with surrogate approximation of constraints. *J. of Global Optimization*, 69(1):117–136, September 2017.

[70] S. Roy. *A Mixed Integer Efficient Global Optimization Framework - Applied to the Simultaneous Aircraft Design, Airline Allocation and Revenue Management Problem*. PhD thesis, West Lafayette, IN, USA, 2017.

[71] S. Roy, K. Moore, J. T. Hwang, J. S Gray, W. A. Crossley, and J. Martins. A mixed integer efficient global optimization algorithm for the simultaneous aircraft allocation-mission-design problem. In *58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 1305, 2017.

[72] S. Roy and W. A Crossley. An EGO-like optimization framework for simultaneous aircraft design and airline allocation. In *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 1659, 2016.

[73] N. Kolencherry and W. A. Crossley. Multi-fidelity optimization strategies using genetic algorithms and sequential kriging surrogates. In *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, Nashville, TN, 2012. AIAA.

[74] MathWorks. Documentation optimization toolbox fgoalattain. 2017.

[75] MathWorks. Documentation optimization toolbox fmincon. 2018.

[76] H. Song, K. K. Choi, and D. Lamb. A study on improving the accuracy of kriging models by using correlation model/mean structure selection and penalized log-likelihood function. In *10th World Congress on Structural and Multidisciplinary Optimization*, Orlando, FL, 2013.

[77] F. A. C. Viana. Surrogates toolbox user's guide, version 3.0 ed., 2011.

[78] G. Jekabsons. RBF: radial basis function interpolation for MATLAB/OCTAVE, version 1.1 ed., 2009.

[79] L. A. McCullers. FLOPS, software package, ver. 8.12. NASA Langley Research Center, 2010.

[80] NASA. NASA Fundamental Aeronautics Program. [Online], 2008.

[81] L. A. McCullers. Flight optimization system, release 8.12, user's guide. NASA Langley Research Center, 2010.

[82] P. Wong and M. Maina. Studies of methods and philosophies for designing hybrid laminar flow wings. In *ICAS 2000 Congress*, 2000.

[83] P. C. Arcara Jr., D. W. Bartlett, and L. A. Mccullers. Analysis for the application of hybrid laminar flow control to a long-range subsonic transport aircraft. 1991.