ABSTRACT

Title of Thesis: ONLINE SURROGATE-BASED MULTI-OBJECTIVE DESIGN OPTIMIZATION USING GENERATIVE ADVERSARIAL NETWORKS WITH CONSTRAINT ASSISTANCE

Arko Chatterjee
Master of Science, 2025

Thesis Directed by: Professor Shapour Azarm
Department of Mechanical Engineering

Associate Professor Katrina Groth
Department of Mechanical Engineering
Center for Risk and Reliability

Multi-objective design optimization problems can be computationally expensive, such is the case with many engineering optimization problems, due to the original objective and/or constraint functions of the problem being costly to evaluate. A method established in current scientific literature to reduce the computational cost for such optimization problems involves the implementation of a surrogate or a lower-cost model to be used in the optimization process in place of the computationally expensive objective/constraint functions. The approach developed in this thesis uses an online surrogate-based optimization method in which the surrogate is developed and iteratively updated as the optimizer converges to a solution.

The primary contribution of this work is the proposal of a new approach for online surrogate-based multi-objective design optimization using generative adversarial networks. A constraint

boundary-informed support vector machine facilitates the approach to predict whether the generated solutions are feasible or infeasible. The performance of the proposed approach is evaluated and compared to two other methods from the literature. The comparison of these methods is carried out using several quality metrics and using numerical and engineering test problems. The engineering test problem is based on the optimization of the operating conditions of an unmanned surface vessel. The results from these test problems indicate that the proposed approach is able to outperform the other approaches for most of the quality metrics and test problems.

ONLINE SURROGATE-BASED MULTI-OBJECTIVE DESIGN OPTIMIZATION USING GENERATIVE ADVERSARIAL NETWORKS WITH CONSTRAINT ASSISTANCE

by

Arko Chatterjee

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2025

Advisory Committee:
Dr. Shapour Azarm, Chair/Advisor
Dr. Katrina Groth, Co-Advisor
Dr. Jay Lee

Foreword

Two publications are used throughout this thesis and the content discussed was made by Arko Chatterjee.

Reference [1] documents the proposed approach discussed in this thesis. I am the main author on that paper.

References [1] and [2] documents aspects of the case study in Chapter 4.

UNIVERSITY OF MARYLAND

2181 Glenn L. Martin Hall
College Park, MD 20742-3035
TEL: 301-405-2410
http://www.enme.umd.edu

4/24/2025

Re: Previously Published Materials appearing in Thesis or Dissertation

Dean of the Graduate School,

Arko Chatterjee (119111727) has:

☐ NOT INCLUDED any previously published works within their thesis or dissertation.

☒ INCLUDED one or more previously published works within their thesis or dissertation. This letter certifies that the examining committee for the student has determined that the student made a substantial contribution to the previously published work. The inclusion of the previously published work has the approval of the thesis or dissertation advisor and the Graduate Director.

Sincerely,

Peter Sandborn
Director of Graduate Studies
Department of Mechanical Engineering
University of Maryland

Sincerely,

Shapour Azarm
Advisor for Arko Chatterjee

Acknowledgments

I would like to thank my advisor Dr. Shapour Azarm, and co-advisor, Dr. Katrina Groth, for their support, guidance, and constructive feedback throughout the course of this thesis. Their technical expertise and mentorship were instrumental in helping me navigate the challenges of this research. I am also grateful to them for offering me a graduate research assistantship to support my thesis. I would like to thank Dr. Jay Lee for being a part of my thesis committee. This study was partially supported by a grant from the Office of Naval Research (ONR). I would also like to thank the academic and technical staff of the Department of Mechanical Engineering at University of Maryland at College Park. Their assistance were vital to the successful completion of this project.

I would also like to acknowledge the U.S. Department of the Navy, Office of Naval Research, who funded, in part, the work presented herein under Grant No. N000142212459. This support does not constitute an endorsement by the funding agency of the opinions expressed in the thesis.

Special thanks to my colleagues and lab partners, particularly Dr. Indranil Hazra, Dr. Ruochen Yang, and Elizabeth Jordan, for the collaborative spirit, troubleshooting sessions, and the many hours spent solving engineering problems together.

I am sincerely grateful to my parents, Drs. Sandip and Devamita Chattopadhyay, and my big sister, Ms. Usoshi Chatterjee, for their unconditional love, support, encouragement, and patience during this journey. Their belief in me provided the motivation I needed to persevere.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

CFD	Computational Fluid Dynamics
CNN	Convolutional Neural Network
DBN	Dynamic Bayesian Network
ECCS	Engine Cooling and Control System
FEA	Finite Element Analysis
GAN	Generative Adversarial Network
GD	Generational Distance
GMOEA	Generative Multi-Objective Evolutionary Algorithm
HV	Hypervolume
No.	Number of
OS	Overall Spread
PCA	Principal Component Analysis
SVM	Support Vector Machine
USV	Unmanned Surface Vessel
WGAN	Wasserstein Generative Adversarial Network
WGAN-GP	Wasserstein Generative Adversarial Network with Gradient Penalty

Chapter 1: Introduction

This thesis is aimed at detailing the proposed online surrogate-based multi-objective design optimizer using a generative adversarial network (GAN) with constraint assistance, with contributions to current literature and providing an understanding of its effectiveness with respect to existing online surrogate-based multi-objective design optimizers. This is done by providing an understanding of all relevant information needed to understand the thesis in Chapter 1. Then, the proposed approach is presented and analyzed in terms of its time complexity in Chapter 2. Chapter 3 demonstrates the effectiveness of the proposed approach on various multi-objective optimization test problems from the literature. The performance of the proposed approach is also compared to other online surrogate-based multi-objective design optimizers using various quality metrics, analyzing how the optimizer performs concerning both closeness to the ideal solution and diversity of the obtained non-inferior solution set. This analysis is continued in Chapter 4, where the proposed approach and other optimizers are applied to an engineering case study for determining the throttle operation control profile for the operation of a USV. Finally, the concluding remarks and future directions of this work will be discussed in Chapter 5.

This chapter aims to provide the introductory material needed to understand the function and purpose of the proposed approach presented in this thesis. This is done by presenting the motivation and objective of this work so that the reader can understand the purpose of this work

and have a general understanding of where this work could be applied. The necessary background information is presented to discuss terminology, methods, and analysis metrics that are brought up throughout the thesis. The primary contributions to the literature are presented to illustrate the explicit goals of this work to the scientific community.

The chapter is organized as follows: Section 1.1 presents the motivation and objective of the study. Section 1.2 details available studies based on a literature review on surrogate-based multi-objective design optimization using generative adversarial network (GAN). Section 1.3 presents information on GAN, support vector machine (SVM), two surrogate-based multi-objective design optimization algorithms (GMOEA and Forrester method), and quality metrics used for comparison of the performance of the optimizers. Section 1.4 explains the differences between the proposed approach and approaches found in the literature. Finally, this chapter ends by describing the organization of the rest of this thesis in Section 1.5.

1.1 Motivation and Objective

Multi-objective design optimization refers to a class of optimization methods that have the goal of finding a set of non-inferior solutions to a multi-objective optimization problem. Non-inferior solutions are the set of solutions where no other point in the set can dominate (perform better in at least one objective without worsening another objective) the given point. This is in regard to a discrete set of values that one would obtain through sampling via a multi-objective design optimizer. Under ideal conditions, this set of non-inferior solutions would have points located within the Pareto front, which would be solutions where no other feasible solutions perform better in at least one objective without worsening at least one other objective for the

problem. However, non-inferior points may not be the same as the Pareto front since the Pareto front is the theoretical best performance that could be achieved for the optimization problem, and in practice, the optimizer may not find these points during the sampling process. Real-world applications of this include developing optimal designs for a variety of engineering challenges.

However, many engineering problems require computationally expensive simulations/calculations. This could come in the form of tuning design parameters for a model that will be run through a computational fluid dynamic (CFD) simulation [3] or performing Finite Element Analysis (FEA) for understanding the physical load on various structures [4]. These are both known to be computationally expensive models as they require a large amount of hardware resources and have a long runtime. So, optimizing the decision variables that govern these problems is impractical using non-surrogate-based design optimizers. This impracticality is caused by non-surrogate-based design optimizers having a high function call requirement [5] (the number of times the optimization problem needs to be evaluated on different sets of decision variables), which results in a high computational load placed on the system. This computational cost in practice would result in long wait times to find the optimal set of decision variables or high computational hardware requirements that can be costly.

This introduces the following research questions:

- What methods are there to solve computationally expensive engineering optimization problems with a limited number of samples?
- Do any of these methods prioritize sampling feasible points in a method outside of just the fitness function?
- Are there any techniques to incorporate a feasibility prediction as part of the optimization

process when there is a limited amount of data that can be sampled?

- How can the performance of the method developed in this study be verified and compared against some well-known techniques from the literature?

These research questions will be answered throughout this thesis. This will be done through the discussion of what methods are currently available in the literature (Section 1.2). The discussion of the limitations of each of the approaches that are currently seen in the literature. The considerations were made in the development of the approaches from the literature. Followed up with a discussion of how the proposed approach can expand upon what is currently seen in the literature (Section 1.4). This expansion is detailed in Chapter 2, with its effectiveness showcased in Chapters 3 and 4.

Upon examining the problem of solving computationally expensive multi-objective design optimization problems, two possible approaches start to emerge for solving such engineering optimization problems. One approach involves reducing the computational load of the optimization process. This could be done by generating an offline surrogate model for the computationally expensive component of the optimization problem, as shown in [6]. This approach approximates the engineering optimization problem by utilizing previously collected data. The other way of solving these computationally expensive engineering optimization problems is to reduce the number of function calls the optimizer needs to use to find optimal solutions. This can be done using an online surrogate-based design optimizer as shown in [7]. The number of function calls is reduced by creating and updating a surrogate model for the optimization problem and sampling where the non-inferior/Pareto solutions are likely located.

In addition to this, there are different methods for determining the interaction with the

underlying surrogate model. This can be done by having the surrogate model approximate the overall optimization problem [8]. Another method is to have multiple surrogate models, where each surrogate approximates a function (each computationally expensive objective function/each constraint function) from the optimization problem [9]. Various model types could be used for both offline surrogate models and online surrogate models. This would include models such as Radial Basis Function [10], Kriging model [11], Support Vector Regression (SVM) [12], Neural Networks [13], Variational Autoencoders [14], and others [15].

Since offline surrogate-based design optimizers require all the data to be collected before the optimization process can start, this requires a diverse and comprehensive representation of the entire design space. In contrast, online surrogate-based design optimizers gather data dynamically during the optimization process, allowing subsequent iterations to focus computational resources on regions of the design space where non-inferior or Pareto solutions are predicted to exist.

The goal of this study is to present a new multi-input, multi-output surrogate multi-objective design optimization method that leverages GANs [16] and SVM constraint models in the optimization process. The proposed approach will be evaluated based on its proximity to a reference set of non-inferior or Pareto solutions, as well as the diversity of the obtained non-inferior solutions.

To develop a detailed understanding of the approaches that currently exist to solve this problem, a comprehensive literature review was conducted. The next section will discuss the structure, findings, limitations, and noticeable gaps in the literature.

1.2 Literature Review

The goal of this literature review is to provide an overall understanding of the work that has been done regarding online surrogate-based multi-objective design optimization using generative adversarial networks. The goal is to focus in specific regards into methods that use a form of constraint assistance to influence the generator's component of the GAN to create feasible solutions outside of the fitness function evaluation for the sampled point. This is associated with the idea of adding additional knowledge to the GAN in a specific regard to the feasibility of the decision variable values generated.

A study of the current literature was conducted through Google Scholar as the primary source to identify relevant literature related to online surrogate-based multi-objective design optimization using generative adversarial networks. In addition to this, many offline surrogate-based multi-objective design optimization algorithms were analyzed to expand the breadth of work reviewed. The guiding keywords that were used for conducting this study include: online surrogate-based optimizers, online approximation-assisted optimizers, multi-objective optimizers using generative adversarial networks, and surrogate-based optimizers using generative adversarial networks with constraint assistance.

Based on the current literature, in recent years, generative models such as GANs have been used for the development of online surrogate-based design optimizers, such as in [15] and [17]. The reason why GANs have been used for these applications is due to the GAN's ability to mimic a distribution of points and their underlying characteristics with a limited amount of data. These points are created through the generator component of the GAN. The background information for how a traditional GAN works is needed to understand the structure of the proposed approach

being discussed in this thesis. More information about traditional GANs is described in Section 1.3.2.

The usefulness of using a GAN for optimization problems is shown in [18]. The authors present a method that is based on a traditional GAN structure but uses the discriminator component from the GAN to assess the fitness function value for a given set of sample points. The generator component of the GAN is used for data augmentation to synthetically increase the amount of data that is used for training. Zhang et al. [18] then uses this as a surrogate model that has been trained on offline data (a pre-collected dataset outside of the optimization process) in an evolutionary optimizer to find the Pareto front solutions, making this approach an example of an offline surrogate-based optimizer that uses a GAN. The limitations of this work are that this method relies heavily on the quality of data found from the initial, offline sampling. This limitation is evident in all offline surrogate-based design optimizers since the surrogate model does not get updated if a decision variable that was not in the original dataset is sampled by the optimizer.

Another example of GANs being used for offline surrogate-based design optimization is in [19]. This paper uses a GAN with polynomial chaos expansion embedded into the generator to create a surrogate that reproduces realistic data from the simulation model. In the paper [19], it states that once the surrogate has been trained, it is then used in a metaheuristic optimizer, such as Particle Swarm Optimization, to find the optimal set of decision variables. The limitations of this work are that the data needs to be generated before the optimization process begins. This means that the solution discovered would need to be validated with additional samples to compare the predicted values provided by the approach and the actual values from the engineering problem. The approach shown in [19] involves having the GAN surrogate model be trained for

each objective and constraint function for the optimization problem. This could prove to be computationally expensive for larger optimization problems (optimization problems with many objective and constraint functions).

An example of GANs being used for online surrogate-based design optimization is in the GMOEA [17]. This method uses a traditional GAN for offspring generation. The decision variable values and the fitness values of the sampled points were used to train the GAN. The generator component of the GAN is used to generate non-inferior offsprings that aim to have a better-performing fitness value than the points that have been sampled so far. The discriminator component of the GAN is used to influence the generator to generate points with characteristics of the non-inferior solutions. The limitations of this work are the repetitive training process for the GAN as it updates with the new information. The implementation of the GAN is susceptible to mode collapse due to the underlying standard GAN model used, which results in a lack of diversity in generated solutions after long training durations/many optimization iterations. The knowledge the GAN uses for training is limited to what is provided solely through the fitness function and contains no explicit constraint handling method outside of the fitness function.

Another example of how GANs are used for online surrogate-based design optimizers is shown in [20]. This was used to solve large-scale multi-objective optimization problems involving hundreds to thousands of decision variables. This is solved using the GAN to map a low-dimensional latent space to the high-dimensional decision variables in the optimization problem. Then, an evolutionary multi-objective optimizer can solve the optimization problem in the lower-dimensional latent space. The limitations of this work concern the amount of information that can be derived from the fitness values of the set of sampled solutions. This means that the surrogate model is only able to make considerations in regards to the predicted

overall fitness of a given set of decision variables but is unable to evaluate any characteristics for how it performs on any specific objective/constraint function. This implementation presents a traditional GAN model and is susceptible to mode collapse in a way that is similar to the approach shown in [17]. This paper does not present any application to any optimization problems that have a real-world parallel and is only tested on test problems from the literature. As a result, the application of this method to complex real-world engineering problems is not known from the information provided.

Due to the GAN's usefulness, different variations of GANs have been incorporated into online surrogate-based design optimizers. This would include methods such as Wasserstein Generative Adversarial Networks (WGANs) [21]. This has been proven to be useful for representing 3D material structures along with their properties [22]. The advantage of using a WGAN instead of a traditional GAN is improvements to the training stability, making it less likely to have issues such as mode collapse. The limitations of this work are the lack of an explicit constraint handling method, and it relies on using gradient descent for the optimization process without any additional features/functions to promote exploration. This would make the algorithm susceptible to only being able to find the local optimum for a non-convex optimization problem and could miss the global optimal solution.

Another variation of GANs is a Wasserstein Generative Adversarial Networks with Gradient Penalty (WGAN-GP), which has been used in protein-peptide docking [23] since it makes stability improvements to the standard GAN model. The limitations of this work are that the knowledge used for training the WGAN-GP model is limited to the fitness information of the sampled points without incorporating any other domain knowledge into the training process.

In addition to the implementation of various types of GAN models, GANs have been used

to improve the accuracy of other surrogate models. This was shown in [13], where a GAN was used to improve the accuracy of a recurrent neural network/multilayer perceptron that was tasked to create accurate depictions of airfoil designs. The limitations of this work are that the implementation shown in this paper performs the optimization offline after all the data has been collected. This limits the accuracy of the resulting solution and requires the initial sampling to be diverse and at a high enough resolution for the surrogate model to be accurate across the entire design space, since there is no knowledge of where the optimal solution is located.

An online surrogate-based multi-objective design optimizer is shown in the Generative Multiform Bayesian Optimizer [24] paper. This method involves using a GAN to transform a large-scale, high-dimensional space into multiple smaller latent spaces and performing optimization in all of these latent spaces simultaneously using a Bayesian optimizer. By performing the optimization in this way, it is easier for the optimizer to converge to a solution in a lower-dimensional space and convert that to a higher-dimensional representation of the decision variables than it is to converge to a solution in a higher-dimensional space. The limitations of this work are that there is no well-constructed rule for the number of lower-dimensional spaces to generate for any multi-objective optimization problem. Since this approach converts the higher-dimensional space to a lower-dimensional representation, errors/inaccuracies could arise in the conversion process due to the GAN model not achieving complete accuracy across the design space. This error could occur in cases where the lower-dimensional representation overlooks the global optimal solutions. This would be the case where the low-dimensional space does not align well with the high-dimensional space.

In terms of online surrogate-based multi-objective design optimization using generative adversarial networks with constraint assistance, the only similar work that was found, to the

author's knowledge, is [25]. In this paper [25], the GAN is not only used to generate offspring, but the discriminator component of the GAN is used to predict if the input data would result in a feasible solution. By embedding this prediction into the discriminator aspect of the neural network, this would have the GAN simultaneously learn to generate points that are feasible and non-inferior. The limitations of this work are that the discriminator is predicting the feasibility of a set of decision variables as part of a neural network. If there is a limited number of data points, there could theoretically be an infinite number of lines/curves that divide the region between being predicted to be feasible or infeasible. Due to this possibility, finding the boundary line would be done through incremental improvements, assuming the discriminator has complete accuracy in separating the two regions.

1.2.1 Gaps/Limitations in Existing Literature

The main gaps/limitations that were observed throughout the current literature that this work fills in is: taking into consideration additional domain knowledge for a general constrained multi-objective optimization problem. Most of these methods did not consider modeling the predicted feasibility of the constraints. These papers showed a tendency to model each objective/constraint problem in the multi-objective optimization problem and run the optimization algorithm on the values predicted for each of the surrogate models, or the surrogate was used to model/predict the fitness of a given set of decision variables and generate points that would have a high performing fitness value. Depending on the fitness function used, this could result in cases where some infeasible points could have a better fitness value compared to a feasible solution due to there not being a strong enough penalty component. By generating points within the feasible domain,

this allows for a more focused exploration in the region where the non-inferior solution set must be located. In the case where the predicted feasibility was considered, it was treated as an aspect of the discriminator. This discriminator component would not produce the boundary line in the location where the maximum separation between the feasible and infeasible solutions is located. By creating a predicted boundary line at the location where maximum separation between the two regions occurs, this could provide a more efficient search strategy for finding the true feasible/infeasible boundary for a general constrained multi-objective optimization problem.

1.3 Background

This section provides the background information for information relevant to the structure of the proposed method, other online surrogate-based design optimizers used to compare performance, and quality metrics to provide quantitative evaluations of the optimization approaches. This will be covering the following topics: GANs, Evolutionary Multi-Objective Optimization Driven by Generative Adversarial Networks (GMOEA), the Forrester method, and quality metrics such as generational distance (GD), Hypervolume (HV), Overall Spread (OS), and Deb's Spread metric (spread).

1.3.1 Constrained Multi-Objective Optimization Problem

The general structure for a constrained multi-objective optimization problem used throughout this thesis is shown in Equation 1.1.

$$\begin{aligned} &\min f_1(\mathbf{x}), ..., f_i(\mathbf{x}), ..., f_n(\mathbf{x}) \\ &s.t. \\ &g_j(\mathbf{x}) \leq 0 \; j = 1, ..., J \end{aligned} \tag{1.1}$$

The different objectives that make up the multi-objective formulation are shown through $f_1(\mathbf{x})$, $f_2(\mathbf{x})$, $f_n(\mathbf{x})$. Where the optimization problem would have n different objectives. The optimization problem has m constraints, which are represented by the $g_1(\mathbf{x})$, $g_2(\mathbf{x})$, $g_m(\mathbf{x})$. Each of these constraints is modeled in a way where the constraint function evaluation must be zero or negative to be feasible, and a positive value would make the given constraint infeasible. $\mathbf{x}$ represents the set of decision variables that the optimization problem is evaluating. This set of decision variables ($\mathbf{x}$) is a vector of n (number of decision variables needed to perform a single evaluation of the problem) components.

Equality statements were not considered due to the limited number of available multi-objective optimization problems with equality constraints, and the equality constraint could not simply be inserted as a component of the objective function. In addition to this, any equality constraint could be separated into two inequality constraints, as shown below (Equation 1.2).

$$g(x) = 0 \rightarrow g(x) \leq 0, -g(x) \leq 0 \tag{1.2}$$

For both $g(x) \leq 0$ and $-g(x) \leq 0$ to be satisfied, $g(x)$ must be zero. For all other values of $g(x)$, at least one of those two constraints would be infeasible.

1.3.2 Generative Adversarial Networks

A detailed description for understanding how GANs work was shown in [16]. It [16] describes that traditional GANs are comprised of two competing neural network models: the generator and the discriminator. The purpose of the generator is to create data that the discriminator would predict to be real. The purpose of the discriminator is to differentiate between real data (from a source outside of the neural network model/not created by the generator) and data generated by the generator. The input to the discriminator consists of either real data or synthetic data produced by the generator. This is done by inputting the generator with white noise data in a latent z space. The generator would transform this into values that would make the discriminator think that the data received belonged to the real dataset. When the random white noise data is provided to the generator, the generator transforms this input data into a distribution that encompasses the real data that has been collected and used for the training process. As the GAN is trained, these models will compete with each other, with the desirable end goal being that the generator can create data that is indistinguishable from potentially real data. The training process of this traditional GAN with two competing neural networks is shown in Equation 1.3.

$$\max_{D} \min_{G} V(G, D) \tag{1.3}$$

where $V(G, D)$ is a value function dependent on the generator, G, and the discriminator, D. This value function is defined in Equation 1.4.

$$V(G, D) = \mathbb{E}_{p_{data}} log(D(x)) + E_{p_g x} log(1 - D(x)) \tag{1.4}$$

where, $\mathbb{E}_{p_{data}} log(D(x))$ represents the discriminator's ability to categorize the real data correctly and $E_{p_g x} log(1 - D(x))$ represents the discriminator's ability to categorize the data generated by the generator as fake. When implementing this in a software implementation, the binary cross-entropy loss (BCE_{loss}) function (Equation 1.5) is used.

$$BCE_{Loss}(x_n, y_n) = y_n \times \log(x_n) + (1 - y_n) \times \log(1 - x_n) \tag{1.5}$$

where y_n is the label (real or fake) associated with the given input sample (x_n). y_n is assigned a value of one if the true label for the given input (x_n) is real and a value of zero if it is fake. Ideally, the discriminator would be trained on a fixed generator (a generator model using fixed values for the weights defining the neural network model, the generator's values are not being updated while the discriminator's weight values are) until it can perfectly categorize the real and fake data. Then the generator would be trained until it reaches its ideal state when the optimal discriminator for a generator has an accuracy of 0.5. This indicates that the generator reaches optimality when the optimal discriminator is not able to distinguish between the real data and the fake data created by the generator. Once the generator has finished training, this process repeats itself with the discriminator training using the new generator. This process would continue until both neural network models converge to a fixed set of weight values for both the generator and discriminator models. This is indicated by the performance of both the discriminator and generator not showing significant improvement between transitions.

In practice, training both the generator and the discriminator until they converge to ideal states (as described above) is computationally expensive and time-consuming. Instead, they are each trained for a limited number of iterations, and the repetition of the transition between

training the discriminator model and training the generator model is also limited. The limitations put in place for this training process are determined by the user depending on the complexity of the dataset, the computational resources available, and the desired quality of the generated data from the GAN.

1.3.3 Support Vector Machine

Another relevant technique that is essential for understanding the underlying mechanics of the proposed approach shown in this thesis is the Support Vector Machine (SVM). This surrogate modeling method, as defined in [26], shows that SVMs are used to create a hyperplane that would divide two different categories of points. These two different regions could be divided by an infinite number of lines. An SVM model finds the hyperplane that results in the maximum distance of separation between the two regions. The properties of the characteristics of an SVM are shown in Equation 1.6.

$$\begin{aligned} wx_i + b \geq 1 \rightarrow y_i = 1 \\ wx_i + b \leq 1 \rightarrow y_i = -1 \\ y_i(wx_i + b) \geq 1 \forall i \end{aligned} \tag{1.6}$$

where, w represents the weights of the SVM, x_i represents the input vector provided to the SVM to make a prediction, y_i represents the predicted label (which is given a value of positive or negative one) for a given input vector, b represents the bias given to the prediction equation. The third line of this equation (Equation 1.6) is used to show that all the points have been correctly characterized. If the true value for a given input vector is one ($y_i = 1$), then the predicted value

$(wx_i + b)$ would be a positive one to maintain this constraint. Likewise, if the true value for a given input vector is negative ($y_i = -1$), then the predicted value $(wx_i + b)$ would be a negative one. Multiplying the true value and predicted value (-1×-1) results in a value of positive one for the constraint. This accuracy constraint must hold for all values of input vectors and their true value outputs. A hyperplane that would result in these properties being true would correctly characterize all the input data. To ensure that this hyperplane functions as an SVM by maximizing the distance between two regions, an additional step is required to maximize the margin between the points. This maximization of the margin between the two regions is shown in Equation 1.7.

$$\min_{x \in D} \frac{|x \times w + b|}{\sqrt{\sum_{i=1}^{d} w_i^2}} \tag{1.7}$$

Equation 1.7 follows the same variable representation as equation 1.6. Once the weights (w) and bias (b) terms have been solved per Equations 1.6 and 1.7, the SVM model would be trained. The trained SVM model can then make predictions for any given input vector (x) provided to the model using Equation 1.8.

$$f(x) = sgn(wx_i + b) \tag{1.8}$$

Additional modifications can be made to the SVM structure by including kernel functions. Kernel functions can be used to convert the input space into a latent space where it would be easier to handle nonlinearities that make the regions non-linearly separable in the input space. This would be used in place of the x_i term in the above equations (Equations 1.6, 1.7, and 1.8). An example of a kernel function is the Gaussian radial basis function shown in Equation 1.9.

$$K(x, x') = \exp(-\frac{||x - x'||^2}{2\sigma^2}) \tag{1.9}$$

where x represents the given input vector, x' represents the center values for the radial basis function (these centers usually correspond to the trained data), σ controls the spread/width of the Gaussian function, with a larger value corresponding to a wider influence of a given trained point.

1.3.4 GMOEA

GMOEA is a gradient-free (where no explicit knowledge about the underlying functions is required as input) surrogate-based multi-objective design optimization algorithm [17] and is used for comparison with the proposed approach. GMOEA modified the use of a GAN not just by using it to create data points similar to the real (sampled) data but by using it to generate points that would result in a high-performing fitness value based on its fitness function for evaluating the quality of a given point The loss function that is used to train this GAN model followed a traditional structure (described in Section 1.3.2) of using a binary cross-entropy loss [27] (Equation 1.5).

In GMOEA, the real/sampled data was compared to a value of zero or one, based on the fitness value for the given point. The fitness function implemented in the GMOEA algorithm is from the improved Strength Pareto Evolutionary Algorithm (SPEA2) [28]. This is a non-surrogate-based design optimization algorithm with a fitness function that values the number of other points a given point dominates (strength) and uses the pairwise Euclidean distance calculation for a given point to all other points to estimate density. This fitness function incorporates aspects of closeness to an ideal point (through the strength component) and diversity (through the

density estimation).

A penalty-based approach was applied since GMOEA does not have an explicitly stated constraint-handling method incorporated into the approach. The penalty is calculated by the total of constraint violations (CV) across all constraints that occurred for the given sample. The CV is calculated using the value of the constraint with a lower bound set at 0. This is shown in Equation (1.10), where $G(x)$ is the value of the constraint equation. This penalty was added to each of the objectives.

$$CV = max(0, G(x)) \tag{1.10}$$

1.3.5 Forrester Method

Another online surrogate-based design optimization algorithm that was used for comparison is the Forrester algorithm [11], which uses a Gaussian Regression Model as a surrogate to estimate the value of the objective functions. The model also provides an estimate for the standard deviation around the predicted value, where the true value could be. This is a measure of uncertainty that changes based on the amount of sampled information close to a given point and the variation of the data at that point. These two metrics (the predicted value and the standard deviation around the predicted value) can be combined to provide an estimate for the maximum expected improvement for a given set of decision variables. The maximum expected improvement is found using an evolutionary optimizer. This is done since evolutionary optimizers do not require gradient information for the underlying optimization problem.

Once the surrogate has been trained using all available data, an optimization algorithm

can be applied to the surrogate model to find the set of decision variables that result in the highest expected improvement value. The set of decision variables where the highest expected improvement value is found becomes a part of the offspring to be sampled in the next generation, improving the accuracy of the model in the area where the optimal value is likely to be found. The surrogate model is then retrained with the new information. No explicit constraint handling method was stated to be used for this method. As a result, the same penalty-based approach used for the GMOEA implementation, of adding the constraint violation as a penalty to the objective evaluation, was applied here.

Since no fitness function for combining the objectives was explicitly stated to be used for this method in [11], the objectives were combined using various weights. This combinatorial approach is shown in Equation 1.11.

$$F(x) = \alpha_1 \times f_1(x) + \alpha_2 \times f_2(x) + ... + \alpha_n \times f_n(x) \tag{1.11}$$

where α_i represents the weight given to a specific objective ($f_i(x)$) from the original optimization problem. The total of the weight values is constructed to equal one ($\sum_{i=1}^{n} \alpha_i = 1$). By creating a variety of weight values (α_i) for a given objective, this allows for the possibility of finding different Pareto front solutions. Equation 1.12 represents the overall fitness function implementation for the Forrester method, with a variety of weights (α_i) for each of the objectives.

$$F(x) = \sum_{i=1}^{N} \alpha_i \times f_i(x) + \sum_{j=1}^{M} \max(0, G_j(x)) \tag{1.12}$$

In the above equation, M is the total number of constraints in the optimization problem, N is the total number of objectives in the optimization problem, $f_i(x)$ is the value of i^{th} objective

function, and $G_j(x)$ is the value of the j^{th} constraint function.

1.3.6 Quality Metrics

Four quality metrics are used in this study [29]: Generational Distance (GD) (Equation 1.13), Hypervolume (HV) (Equation 1.14), Overall Spread (OS) (Equation 1.15), and Deb's Spread Metric (Spread) (Equation 1.16). The GD is the distance between the obtained solutions and the reference non-inferior solutions/Pareto front for a given multi-objective optimization problem. The GD (Equation 1.13) [30] was used as a metric to compare the accuracy of the obtained solution to the true solution. The d_i in Equation 1.13 represents the minimum distance between the objective values of the obtained non-inferior solutions and the reference non-inferior front of the problem, and $|Q|$ represents the number of points used.

The obtained non-inferior solutions are the set of points for a given run of a multi-objective design optimizer that cannot be dominated by any other sampled point. The reference non-inferior front is used as a guide to compare the obtained non-inferior solutions using the various quality metrics presented in this section. Since many of the problems presented in this thesis do not have a well-defined Pareto front solution, a reference non-inferior front is used in its place. The reference non-inferior front is a set of non-inferior solutions obtained from running a multi-objective design optimizer until the provided results converged to a solution. In this thesis, NSGA2 [29] was used as the multi-objective design optimizer to create the reference non-inferior solution front. This multi-objective design optimizer was run multiple times, and the non-inferior solution of all points sampled across all runs was used to form the reference non-inferior solution front.

$$GD = \frac{\sqrt{\sum_{i=1}^{|Q|} d_i^2}}{|Q|} \tag{1.13}$$

The lower the GD, the closer the population is to the true Pareto front solutions or the reference non-inferior solutions.

The HV is the total area/volume that the obtained non-inferior solutions can dominate. The HV (Equation 1.14) is used to calculate the total volume that is created from a reference point. This calculation is done by calculating the overall volume filled by all of the individual volumes (v_i) between each point and the reference point, which is the nadir point for the problem.

$$HV = volume(\cup_{i=1}^{|Q|} v_i) \tag{1.14}$$

A larger HV indicates that the non-inferior solutions are farther from the reference point. The largest possible HV for a given reference point is created when the non-inferior points are the same as the true Pareto Front. So, having a larger HV indicates that the non-inferior solutions are likely closer to the true Pareto front for the optimization problem.

The OS (Equation 1.15) is used to evaluate how the spread of the non-inferior solution set compares to the reference non-inferior front. This is calculated based on the volume of the space explored by the extreme points of the non-inferior solution set within the global boundary ($HR_{ex}(P)$) and the global boundary for the problem (HR_{gb}), which is derived from the true solution of the problem.

$$OS = \frac{HR_{ex}(P)}{HR_{gb}} \tag{1.15}$$

A larger OS indicates that the extreme points of the obtained non-inferior solution encapsulate more area/volume of the actual area/volume created by the reference non-inferior solutions/Pareto front. This shows how any point that could fall within the area/volume of the obtained non-inferior set of solutions lines up with the area/volume of any point of the reference non-inferior solutions/Pareto front for a given multi-objective optimization problem.

The spread (Equation 1.16) was used to measure the diversity of the non-inferior solutions. Spread is calculated by comparing the distance between adjacent points (d_i) to the average distance between points ($\bar{d}$), as well as the distance between extreme points (d_m^e) (best and worst points in terms of the different objectives) to the extreme points of the true Pareto front. A lower spread metric indicates more diversity among the non-inferior solution set. This can be thought of as a measure of how uniformly distributed the obtained non-inferior solutions are for a given multi-objective optimization problem.

$$\Delta = \frac{\sum_{m=1}^{M} d_m^e + \sum_{i=1}^{|Q|} |d_i - \bar{d}|}{\sum_{m=1}^{M} d_m^e + |Q|\bar{d}} \tag{1.16}$$

1.4 Differences between Proposed Approach and Existing Methods

This section will explain the differences between the proposed approach and approaches found in the current literature in terms of online surrogate-based multi-objective design optimizers. To reiterate, the proposed approach is for a new online surrogate-based multi-objective design optimizer using GANs with constraint assistance. The proposed approach uses the GAN to create offspring sample points for the optimization process. The GAN model is influenced by both the fitness evaluation of the previously sampled decision variables and information obtained from the

SVM constraint surrogate models. Both the fitness evaluation and the SVM constraint surrogate models are used to augment the loss function in the GAN's generator model in an attempt to have the GAN generate points that have a high likelihood of being feasible and within the desirable region of the non-inferior solutions.

The differences between the proposed approach and typical online surrogate-based multi-objective design optimizers, such as that shown in the Forrester method [11], are that the surrogate is used (in the Forrester method) to model either the individual objective/constraint functions or the overall fitness of the design space based on the sampled points. These methods require the optimizer to find possible sets of decision variables that would be within the reference non-inferior solution set (ideally on the Pareto front) for the problem. In the case of approaches similar to the Forrester method, the possible sets of decision variables are discovered through the use of an additional optimizer (commonly an evolutionary optimizer). Since the proposed approach uses a generative model to find the optimal solutions, the result of the GAN is to model a distribution/entire region of the design space where the reference non-inferior solution set (ideally, the Pareto front) could be located. The underlying optimizer used for approaches similar to the Forrester method could focus too much on a specific area too quickly. By using a generative modeling-based approach, the proposed approach can avoid the pitfalls of requiring a high level of accuracy for a large initial sampling set to train the surrogate model.

Other differences between the proposed approach and other online surrogate-based multi-objective design optimizers that use generative models (such as GANs), is shown in [17], are that the surrogate GAN model is used to generate a distribution of points where the non-inferior set of solutions are likely found through the use of the calculated fitness values of the sampled points. While the proposed approach involves the GAN modeling, the overall fitness of the design

space, additional considerations are made to sample points within the predicted feasible region, in addition to the desirable region created based on the fitness values of the sampled points. The advantage this creates for the proposed approach over the approach shown in [17] is that the exploration and exploitation performed by the optimization process through the GAN is focused within the region that has the highest predicted likelihood of producing a feasible solution. So, for practical implementations, an infeasible result might be completely unacceptable. Using this method would ensure that more of the samples come from the region that is predicted to be feasible. This could allow for a temporary implementation that is not optimal but produces sufficiently good enough performance while more evaluations can be conducted.

There have been cases presented in the literature where the online surrogate-based multi-objective design optimizer was able to model the distribution of possible Pareto front points through generative modeling and made considerations for the constraint feasibility predictions, such as shown in [25]. The advantages stated in this approach from the literature [25] are in terms of the convergence to the obtained non-inferior set of solutions and improvements with diversity. This is done by separating the sampled points from the original optimization problem into feasible and infeasible sets. The discriminator is trained to predict which set a given set of decision variables belongs to. This discriminator model is then used for the training of the generator. The generator in [25] is used to generate the offspring solutions, which are expected to be feasible and would perform well based on a fitness function incorporated into the optimization algorithm.

There are limitations in the method used to model the constraint feasibility, as described in Section 1.2. The proposed approach models the constraint through a separate SVM model for each constraint. This allows for the focus of the constraint assistance model to be only on

the feasibility of an individual constraint. This also allows the constraint surrogate model to be simpler and more generalizable than a model that must model the feasibility of a given set of decision variables and its likelihood of being within the desirable region of the design space.

1.5 Contributions to the Literature

The contributions stated in this section were also presented in [1], which has been submitted to the Journal of Mechanical Design (JMD). The primary contributions to the literature are as follows:

1. Developed the structure for a new online surrogate-based multi-objective design optimization algorithm that uses a GAN-based framework aided by a support vector machine (SVM) to find feasible points for black-box computationally expensive multi-objective optimization problems. The SVM surrogate model estimates the probability of a constraint violation occurring at a given point. The trained SVM is used to influence the GAN to generate offspring within the regions that have a higher likelihood of being feasible for all the constraints in the optimization problem. Existing surrogate-assisted multi-objective design optimization methods use GANs in various ways. Some methods from the current literature use the GAN to approximate individual objective/constraint functions [25], while others approximate the fitness values for various points [17], [31]. Some approaches incorporate the constraint handling as an aspect of the discriminator [25]; however, this does not provide the maximum separation between the feasible and infeasible regions of the design space where the true boundary can be assumed to be. The proposed structure uses an SVM to define the constraint boundaries to create the separation between the feasible and

infeasible regions, as this is a built-in feature to the training process for an SVM [26]. By incorporating this into the loss function of the GAN, the GAN can generate solutions that have a higher likelihood of being feasible.

2. Evaluated the performance of the proposed constraint-assisted generative online surrogate-based multi-objective design optimization approach using quality metrics such as generational distance (GD) (Equation 1.13), hypervolume (HV) (Equation 1.14), overall spread (OS) (Equation 1.15), and Deb's spread metric (spread) (Equation 1.16). These metrics were used to compare the proposed approach to two existing online surrogate-based design optimization algorithms from the literature. This provides a broader understanding of how the different algorithms perform in terms of accuracy and diversity, whereas many other papers that have compared their algorithms to online surrogate-based design optimization algorithms, e.g., [17] [32], focus solely on the accuracy of the results using metrics such as GD and HV.

3. Applied the proposed constraint-assisted generative online surrogate design optimization approach to a wide variety of numerical test problems from current literature and an engineering case study for the operation of an unmanned surface vessel (USV). This demonstrated the scalable performance of the proposed approach. Throughout these optimization problems, the proposed approach was compared to other methods, constraining all approaches to the same number of allowable function calls. Finally, an estimate for the time complexity of the proposed approach was determined and used for comparison to the time complexity obtained for the other methods tested.

1.6 Organization of Thesis

The rest of this thesis is organized using the following structure. Chapter 2 discusses the structure of the proposed approach for a new multi-input multi-output online surrogate-based multi-objective design optimizer. Chapter 3 demonstrates the effectiveness of this proposed approach on a set of test problems from the current literature. Then, Chapter 4 shows the effectiveness of this proposed approach on a case study problem for determining the throttle control profile for an unmanned surface vessel. Finally, the conclusion to this thesis includes the summary of the work done, contributions to the literature, limitations of this work, and possible future work that is based on this research and is presented in Chapter 5.

Chapter 2: Proposed Approach

As discussed in chapter 1, the use of surrogate models has greatly improved multi-objective design optimization for computationally expensive optimization problems. Such methods have expanded to include GANs in their surrogate modeling techniques for optimization [17] and [3]. This chapter presents a new multi-input multi-output online surrogate-based multi-objective design optimizer that uses GANs with constraint assistance that are modeled using SVMs. This chapter provides a detailed explanation of the flow of the algorithm through the different steps performed in the optimization process. These steps would include the initial point sampling, constraint model generation, training the GAN model, generating offspring, and the resulting output of the optimizer once the optimization process has finished. The goal of this chapter is to provide a comprehensive understanding of each stage of the proposed approach and their roles in the optimization process. This approach was influenced by the work done in [17] and [28].

This chapter starts with a high-level explanation of the structure of the proposed approach in Section 2.1. Section 2.2 discusses the methodology used for generating the initial population of points. Section 2.3 explains how the SVM constraint models are trained and how they can predict a boundary between the feasible and infeasible regions in the design space. Section 2.4 presents the process for training the GAN using the information from the SVM constraint models and already sampled points with their fitness evaluations. Section 2.5 describes how

the new offspring points are generated from the GAN and selected using the existing sampled points and the surrogate models that have been trained so far for guiding the exploration and exploitation of the design space. Section 2.6 discusses the final output from the optimizer once the optimizer has finished running. Finally, Section 2.7 analyzes and compares the theoretical time complexity of the proposed approach to the theoretical time complexity of two other online surrogate-based multi-objective design optimizers from existing literature (GMOEA [17] and Forrester method [11]).

2.1 Structure of Proposed Approach

A positive integer number of allowable function calls (fcn_{limit}) must be specified before the optimization process can begin. A single function call is a single evaluation of the entire optimization problem. This would require getting all the values for all the objectives and constraints within the optimization problem.

The proposed approach, shown in Algorithm 1 and illustrated in Figure 2.1, starts with sampling an initial set of points (line 1 of Algorithm 1). This evaluation would provide the objective and constraint values for the sampled decision variables. The decision variable values and the constraint values train the constraint surrogate model(s) that are represented as different SVMs (line 3 of Algorithm 1). Each constraint in the optimization problem has an associated SVM model, which is used as the surrogate model to predict if a given set of decision variables will be feasible for a given constraint. This would create a boundary between a predicted feasible and infeasible region. Once all the constraint models have been trained, the algorithm trains the GAN (line 4 of Algorithm 1). A part of the training process of the GAN is to calculate the fitness

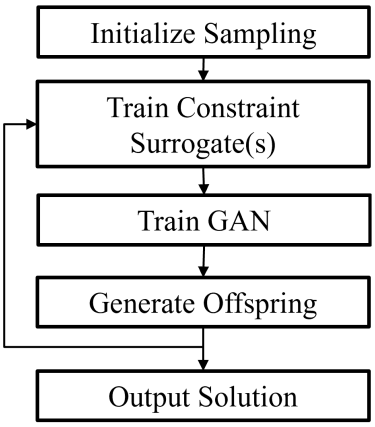

Figure 2.1: Proposed Approach Framework Flowchart

value for each sampled point. The SPEA2 fitness function [28] was used to calculate these fitness values. Then for the training portion of the GAN, the constraint surrogate SVM model(s) and the fitness values of the sampled points are used to train the GAN. The GAN would learn to generate decision variables that would be expected to have a well-performing fitness value and are likely to be feasible.

The generator model from the trained GAN is used to generate decision variable values as offspring for the next generation of points to sample (line 5 of Algorithm 1). The determination of the offspring points to the sample is also informed using the fitness of the sampled points and the constraint surrogate model(s). The process is repeated until the algorithm has used all the allowable function calls. Once completed, the algorithm returns the set of non-inferior feasible solutions in terms of their decision variable, objective, and constraint values (Output Solution).

2.2 Step 1: Initial Point Selection

The initial point selection aims to obtain a diverse understanding of the design space for the optimization problem. This serves as the first step in the algorithm as shown in line 1 of

Algorithm 1 Proposed Approach Framework

Require: $fcn_{limit} \geq 0$

pop,obj,con$\leftarrow$ Initial Point Selection
while $eval < fcn_{limit}$ **do**
    svm_{con}=TrainConstraintModels(pop,con)
    Train GAN(svm_{con},pop,obj,con)
    pop,obj,con=OffspringGeneration(GAN)
end while
    return pop, obj, con

Algorithm 1. The algorithm performs best when it samples both feasible and infeasible points. To achieve this, Latin Hypercube Sampling [22] was used. However, other sampling methods, such as sampling from a uniform random distribution [33], could also be employed.

Since there is a desire to find both infeasible and feasible points during the initial sampling for this algorithm, this can be aided by sampling points along the boundary of the decision variables (the upper/lower bounds for the decision variables). For many problems, such as the TNK problem [34], the boundary values for the decision variables are infeasible ($x_1 = \pi, x_2 = 0 \rightarrow g_1(X) = 8.7696 \geq 0, g_2(X) = 13.956 \leq 0.5$). x_1 and x_2 are the decision variables, while $g_1(x)$ and $g_2(x)$ are the two constraint function values for the TNK problem for a given set of x values (x_1 and x_2). When x_1 is set to the upper boundary value of π and x_2 is set to the lower boundary value of 0, this produces a result that is feasible for the first constraint(g_1) but infeasible for the second constraint (g_2). Both feasible and infeasible points for the different constraints present in the initial sample set are used to create the boundary line between feasible and infeasible solutions by the SVM during the training of the constraint model(s). The feature can be enabled as a hyperparameter for the algorithm. This hyperparameter would be influenced by knowledge of the optimization problem, such as the importance of boundary points, and the number of decision variables.

The goal of this step is to obtain a diverse set of decision variables and their associated objective and constraint values. This should provide the algorithm with a general understanding of the design space for the optimization problem.

2.3 Step 2: Train Constraint Model(s)

Using the sampled decision variables and their associated constraint values, the constraint surrogate SVM model(s) can be trained (line 3 of Algorithm 1, Algorithm 2). This is needed to influence the GAN to generate decision variables that are likely to be feasible. The constraint surrogate model(s) use all the available data that has been sampled from the original optimization problem so far for training (line 2 of Algorithm 2).

Algorithm 2 Training Constraint Models

Require: pop, con, n_{con}
1: **while** $i \leq n_{con}$ **do**
2: $\quad svm_{con_i} = TrainSVM(pop, con_i)$
3: **end while**
return svm_{con}

An SVM using a radial basis function as the kernel ($K(x_i, x)$) is chosen as the constraint surrogate model. The SVM model was used as the constraint model because once an SVM has been trained on a set of values it can calculate the line/curve that would provide the maximum separation distance between the two different categories of points [12]. This line/curve generated by the SVM can be used as an approximation of the boundary line between feasible and infeasible solutions.

This information is provided to the GAN to create offspring decision variable values that have a higher likelihood of being feasible. The specific type of SVM used in this algorithm is

an SVM with a radial basis kernel that would allow it to capture the non-linearity present in differentiating the feasible region from the infeasible region. Once the SVM has been trained, a decision function (Equation 2.1) can be derived from the model. In this decision function (the underlying function for SVM prediction [12]), α represents the Lagrange multipliers, y_i is a binary value indicating if the constraint has been violated, x_i are the support vectors, x are the decision variables to be tested/input values, and b is a bias term added to the function.

$$f(X) = \sum_{i=1}^{N} \alpha_i y_i K(x_i, x) + b \tag{2.1}$$

This decision function (Equation 2.1) is passed onto the GAN to predict where the feasible points are located. This process for training the constraint surrogate model is repeated for each constraint present in the original optimization problem (lines 1 and 3 of Algorithm 2). This shows that the primary impact of the number of constraints in the original optimization problem is that a constraint model must be developed and updated for each iteration of the optimization process for each constraint present in the optimization problem. The output of Step 2 is a trained set of SVMs acting as the constraint surrogate models to approximate the boundary between the predicted feasible and infeasible regions.

2.4 Step 3: Train GAN

A GAN model is used as the optimizer component of the proposed approach, this is the component responsible for generating the offspring decision variables. The reason that a GAN model is used is because of the inspiration from GMOEA [17]. It showed that GAN models were able to perform well for representing a distribution of points that mimic the properties of a region

of a design space. When this is implemented in a multi-objective optimizer, it can perform well with a limited number of sample points. As the optimization process continues, this distribution morphs into the region of the design space that contains the decision variable values that result in the Pareto front.

The process for training the GAN (line 4 of Algorithm 1) uses the constraint surrogate model(s), the sampled decision variables, along with their objective and constraint values. The first step to train the GAN (line 1 of Algorithm 3) is to calculate the fitness value for all the points sampled. This fitness calculation was performed using the fitness calculation from the SPEA2 algorithm [28]. This fitness function consists of two components that calculate the strength and estimate the density for a given point. The strength component is calculated by determining the number of points a given point dominates. The density estimation is calculated using the pairwise distance between the given point and all other points. By including both of these attributes in the fitness function, the fitness calculation takes into account the quality and diversity of the points. This fitness function has been augmented by applying a weight factor to adjust the influence of the density estimation on the calculated fitness value. Once the fitness values for all the sampled points have been collected, the points are sorted based on their fitness value. A subset of the points is selected to represent the desirable region of the design space that the GAN should try to generate points in. This subset of points is retrieved from the points with the best-performing fitness function values and is given labels that the GAN can use for training (line 2 of Algorithm 3).

The GAN is trained for n_{epoch} iterations (line 3 of Algorithm 3). The way that the GAN is able to generate points that fall within the desirable region of the design space is through the use of BCE_{Loss} component (Equation 2.2) in the loss functions for the discriminator (Equation

Algorithm 3 Training GAN

Require: $fcn_{limit} \geq 0$

1: $fit \leftarrow CalcFitness(pop, obj, con)$
2: $y \leftarrow AssignLabels(fit)$
3: **while** $i \leq n_{epoch}$ **do**
4: Calculate Discriminator Loss (Eqn. 2.3)
5: Update Discriminator Weights
6: Calculate Generator Loss (Eqn. 2.4)
7: Update Generator Weights
8: **end while**
return GAN

2.3) and generator (Equation 2.4). The process for training the GAN follows a structure similar to the training process for a traditional GAN [16]. The BCE_{Loss} equation shown in Equation 2.2 is based on Pytorch's BCE_{Loss} function [27] and follows a similar methodology for training that was shown in [17]. As stated in Section 1.3.2, y_n represents the label given to the sampled point (x_n) (1, if the value is from the sampled points and 0, if the value is not from the sampled points or is not within the desirable region of the design space).

$$BCE_{Loss}(x_n, y_n) = y_n \times \log(x_n) + (1 - y_n) \times \log(1 - x_n) \tag{2.2}$$

The loss function for the discriminator component (Equation 2.3) (line 4 and 5 of Algorithm 3) of the GAN used in the proposed approach is the same loss function that was used in GMOEA's implementation [17] of the discriminator component of the GAN. Additionally, the model structure for the discriminator used in the proposed approach followed a similar structure to the structure of the discriminator used in GMOEA's implementation.

$$L_D = BCE_{Loss}(D(x), y) + BCE_{Loss}(D(G(x_{noise})), 0) \tag{2.3}$$

The $D(x)$ represents the predicted value calculated from the discriminator that indicates if the given set of decision variables falls within the desirable design space region when the values of the decision variables that have already been sampled are provided as input. The x_{noise} represents a random noise input that is fed into the generator. These random noise points are generated from a multivariate normal distribution centered on the subset of points that make up the desirable region. The $G(x_{noise})$ represents the predicted values from the generator that are used to estimate possible points that will fall within the desirable region of the design space. The loss function for the generator (Equation 2.4) (lines 6 and 7 of Algorithm 3) contains additional components that are not seen in the training process of a traditional GAN's generator or the training process of the generator in GMOEA's GAN implementation.

$$L_G = \alpha \times BCE_{Loss}(D(G(x_{noise})), 1) + \beta \times \frac{1}{k} \sum \sum SVM_k(G(x_{noise})) \tag{2.4}$$

The α and β terms present in the generator's loss function are hyperparameters used to balance the weight given to the constraint violation prediction loss component and the standard GAN's generator loss component. The $SVM_k(G(x_{noise}))$ is the constraint violation prediction loss component. This is derived from the trained SVMs, discussed in section 2.3, to predict if a given set of decision variables will violate a given constraint. As the generator's training process is performed, this will influence the generator to generate points that are likely in the feasible region of the design space. The $BCE_{Loss}(D(G(x_{noise})), 1)$ component is the standard GAN's generator loss component. As the GAN's training process occurs, this component will influence the generator to generate decision variable points that make the discriminator think that

the provided set of decision variables is within the desirable region. Once trained, the GAN will ideally generate decision variable points that have a low fitness value and are feasible. The result of Step 3 is a trained GAN that can be used to generate decision variables that are likely feasible and would be expected to fall within the desirable region of the design space.

2.5 Step 4: Generate Offspring

After the GAN training process has been completed, the algorithm enters the offspring generation stage (line 5 of Algorithm 1). The process of generating the offspring point(s), which would then be evaluated using the original (computationally expensive) optimization problem, is shown in Algorithm 4. The first step (line 1 of Algorithm 4) of the offspring generation stage is for the generator from the trained GAN to generate a large set of possible decision variables to sample (pop_{off}). Then the feasibility prediction ($feas_{off}$) for each set of decision variables is calculated using the trained SVM constraint model(s) (lines 2 to 5 of Algorithm 4). With higher values of the $feas_{off}$, it is more likely that the constraints in the given set of decision variables will violate at least one constraint in the original (computationally expensive) optimization problem. This calculation is performed as a running total for all the constraints in the optimization problem. The offspring population will only retain a subset of the most feasible offspring decision variable values generated by the GAN (lines 6 of Algorithm 4).

The GAN's trained discriminator is used to evaluate the likelihood of the remaining offspring decision variable points being within the desirable region (line 7 of Algorithm 4). This proves to be an effective way for calculating the predicted quality of the possible sets of decision variables since that was the purpose during the GAN's training process. Using this calculation, a subset of

Algorithm 4 Offspring Generation

Require: GAN

1: $pop_{off} \leftarrow GAN$
2: **while** $i \leq n_{con}$ **do**
3: $\quad feas_{off} = feas_{off} + SVM_{con_i}(pop_{off})$
4: $\quad i = i + 1$
5: **end while**
6: $pop_{off} \leftarrow Top_x(\{x \subseteq pop_{off}\}, feas_{off})$
7: $good_{off} \leftarrow GAN_{Dis}(pop_{off})$
8: $pop_{off} \leftarrow Top_x(\{x \subseteq pop_{off}\}, good_{off})$
9: $dist_{off} \leftarrow DistanceComp(pop_{off})$
10: $pop_{off} \leftarrow Top_x(\{x \subseteq pop_{off}\}, dist_{off})$
11: $pop_{off} \leftarrow Mutation(pop_{off})$
12: $obj_{off}, con_{off} \leftarrow Evaluate(pop_{off})$
return $pop_{off}, obj_{off}, con_{off}$

the best performing points based on the GAN's discriminator score are kept (line 8 of Algorithm 4).

The next step in the offspring generation process is to calculate the pairwise Euclidean distance between the remaining possible offspring decision variables and the decision variable values that have already been sampled (line 9 of Algorithm 4). The best set of points to choose from this set would be the points that have the farthest minimum distance between themselves and the points already sampled so that the algorithm can further explore the design space. So the points with the farthest minimum distance are kept within the set of possible offspring decision variables (line 10 of Algorithm 4). To further incentivize exploration of the design space, a polynomial mutation is used on pop_{off}, before it is evaluated using the actual optimization problem (line 11 of Algorithm 4). Finally, the decision variable values, the objective values, and the constraint values of the offspring sampled points are returned (line 12 of Algorithm 4) and the process continues from step 2 (section 2.3) until all of the allowable function calls have been used.

The resulting output of Step 4 is the new sample point/set of sample points used to update the models (GAN and Constraint Surrogates) with new information including the new decision variable values, objective values, and constraint values.

2.6 Step 5: Output Solution

Once the algorithm has used up all the allowable function calls, it outputs the final set of solutions found. The output information would include the decision variable values, the objective values, and the constraint values for this set of points. The non-inferior set of points from the final set of solutions can be used as the observed (obtained) non-inferior solutions for the optimization problem that is being solved.

2.7 Time Complexity Analysis

Two other online-surrogate multi-objective design optimization approaches were chosen in comparison to the proposed approach. These two approaches are GMOEA and Forrester method. The reason that GMOEA [17] was chosen is that it was the initial inspiration for the proposed approach for using GANs to create offspring sample points to be used in the optimization approach. Forrester method [11] is chosen because it is a well-established online-surrogate optimizer that is in the current literature.

The theoretical time complexity of the tested approaches (proposed approach, GMOEA, and Forrester method) was determined using Big O notation. This analysis evaluates the worst-case scenario for how the algorithm's runtime grows based on various factors for a fixed number of function calls that is kept uniform across the different approaches. Since each of these approaches

is provided with the number of allowable function calls as an input set by the user to the optimizer, this would preset the minimum computational time of the approach to be the number of allowable function calls times the duration it takes to acquire a single sample ($n \times T_{eval}$). For the comparison of the time complexities of the different approaches, this computational cost is excluded from the evaluation so that a comparison on the performance of the approach alone can be evaluated.

The time complexity calculations are calculated in terms of the number of function calls (n), the number of constraints in the optimization problem (n_{con}), the number of weight combinations for the objectives (n_w), the number of training epoches (n_{epoch}) for the GAN, and the number of decision variables (d).

2.7.1 Time Complexity of Proposed Approach

The theoretical time complexity of the proposed approach was determined to be $O(n^2(d \times n_{con} + log(n)) + n_{epoch} \times n(n_{con} \times d + 1))$. This calculation was determined using the main scalable components of the algorithm, which are the process of training the SVM constraint model (Algorithm 2), the GAN training process (Algorithm 3) and the calculation of the fitness values for each of the points sampled in the data set using the SPEA2 fitness function [28] (included in Algorithm 3). An SVM has a known time complexity of $O(n^2 \times d)$ [35]. The known time complexity of making a prediction using a trained SVM constraint model is $O(n_{SV} \times d)$, where n_{SV} represents the number of calculated support vectors obtained from training. The number of support vectors varies as the SVM is solved from generation to generation of the optimization algorithm. However, the maximum number of support vectors an SVM can have is the number of sample points present, so n_{SV} can be set to the upper bound of n ($n_{sv} \leq n$).

The SVM constraint model(s) contributes $O(n^2 \times d \times n_{con})$ to the overall time complexity of the algorithm since a constraint SVM needs to be trained for each constraint in the problem (n_{con}).

The GAN contributes a time complexity of $O(n_{epoch} \times n(n_{con} \times d+1))$. This arises because the trained SVM constraint model(s) perform predictions with a time complexity of $O(n \times d)$ for each constraint (n_{con}) during each training epoch of the GAN (n_{epoch}). Additionally, the GAN also has a training cost proportional to $n_{epoch} \times n$ based on the available data. Also, the theoretical time complexity of the SPEA2 fitness function must be considered, and this was shown to be $O(n^2 \times log(n))$ in [28].

2.7.2 Time Complexity of GMOEA

The theoretical time complexity of GMOEA was calculated based on the code provided for the algorithm. The major scalable computational expense of this algorithm is from the training of the GAN and the SPEA2 fitness function [28]. Since the training process for the GAN in GMOEA is similar to that of the proposed approach, both would have similar computational expense of $O(n_{epoch} \times n)$ from the GAN. Additionally, the SPEA2 fitness function adds a time complexity of $O(n^2 \times log(n))$. The overall theoretical time complexity of GMOEA becomes $O(n_{epoch} \times n + n^2 \times log(n))$.

2.7.3 Time Complexity of Forrester Method

The theoretical time complexity of the Forrester method was calculated based on the implementation of the algorithm inspired by [11]. The Guassian process regression model is the underlying surrogate model that is used in the Forrester method. This algorithm's major scalable computational

expense is in the training of the Gaussian process regression model. Since the implementation was based on a weighted combination of the different objectives, a different Gaussian model was trained for each weight combination. The computational complexity of training the Guassian process regression model is represented as $O(n^3)$ as shown in [36]. This would make the overall time complexity of the Forrester method $O(n^3 \times n_w)$.

2.7.4 Time Complexity Comparison

From this it is shown that in terms of theoretical time complexity GMOEA performs the best. Both GMOEA and the proposed approach can outperform the Forrester method since the highest degree variable in GMOEA and the proposed approach is $n^2 \times log(n)$ whereas the highest degree of the Forrester method is n^3. GMOEA is able to outperform the proposed approach since GMOEA does not take on the computational cost that is generated by the inclusion of the SVMs.

2.8 Summary

This chapter presented a step-by-step explanation of the proposed approach for online surrogate-based multi-objective design optimization. It demonstrated the methodology for integrating GANs and SVMs to learn different aspects of the optimization problem to reduce the computational cost of the optimization problem while making improvements in terms of exploring the feasible region of the design space. The time complexity analysis provides a theoretical understanding of how the proposed approach compares to two established online surrogate-based multi-objective design optimizers. The theoretical time complexity comparison shows that the inclusion of constraint models increases the computational cost slightly compared to GMOEA. However, the

proposed approach still maintains a computational advantage over other optimization methods such as the Forrester method.

The next chapter will evaluate the performance of the proposed approach using various test problems from existing literature. This will include an illustrative example of the optimization process shown in this chapter as it runs through a test problem. The proposed approach will be compared to GMOEA and the Forrester method to show how the proposed approach compares in terms of different quality metrics measuring the closeness and diversity of the obtained feasible non-inferior solutions for each of the methods.

Chapter 3: Test Problems

Evaluating the performance of the proposed approach requires a diverse set of test problems to evaluate how it performs in different conditions. In the case of multi-objective design optimizers, both the diversity of solutions and closeness to non-inferior/Pareto solutions are valuable. So various quality metrics are used to measure the diversity and closeness of the final results generated by the online surrogate-based design optimizers tested. This chapter presents a comparative analysis of the proposed approach against two existing online surrogate-based design optimizers (GMOEA [17] and the Forrester method [11]). This chapter will focus on constrained non-convex multi-objective optimization problems. These are treated as black-box functions where no information about the gradients of the optimization can be calculated. It can be assumed that there are cases where calculating the gradient would be computationally expensive if the evaluation of the optimization problem is computationally expensive. The reason for focusing the evaluation on constrained non-convex multi-objective optimization problems is to evaluate each of the tested approach's ability to generate high-quality, diverse, and feasible non-inferior solutions for their allowable number of function calls in cases where the Pareto front could be non-linear, discontinuous and complex. The goal of this chapter is to assess the performance of the proposed approach in a variety of test problems with varying number of decision variables, objective functions, and constraint functions. This performance is evaluated using multiple

quality metrics to evaluate how close the obtained non-inferior solutions are to the reference non-inferior/Pareto solutions, and the diversity of the obtained non-inferior solutions.

This chapter begins by describing the experimental setup for the different test problems, along with brief descriptions for each of the quality metrics in Section 3.1. Section 3.2 demonstrates an illustrative example with a visual for how the proposed approach functions through the various steps of the optimization procedure shown in Chapter 2. Section 3.3 shows the mathematical formulation and descriptions for each of the test problems. Section 3.4 provides the results and visual comparisons of the best and worst case scenarios for each of the tested approaches for the different test problems. Section 3.4 also contains the quantitative metrics that can be used to compare the performance of the approaches in terms of their closeness and diversity.

3.1 Experimental Setup and Performance Metrics

GMOEA [17], Forrester method [11], and the proposed approach was used to solve various constrained multi-objective problems. The problems tested were TNK [34] (Equation 3.1), OSY [37] (Equation 3.2), CTP [29] (Equation 3.3), and ZDT9 [38] (Equation 3.4). The TNK (Equation 3.1), OSY (Equation 3.2), and CTP (Equation 3.3) test problems were chosen because they are well-established constrained multi-objective optimization problems in the current literature. These test problems have non-convex Pareto Fronts which make it difficult to solve for the global optimum set of solutions (Pareto Front) strictly using methods like gradient descent. The ZDT9 test problem (Equation 3.4) serves as a scalable benchmark for assessing the performance of different algorithms across various decision variables, objectives, and constraints. Ten runs were performed for each test problem to calculate the mean and standard deviation for each quality

metric.

The quality metrics (shown in detail in Chapter 1) used were generational distance (GD) [39] (Equation 1.13), hypervolume (HV) [40] (Equation 1.14), overall spread (OS) [41] (Equation 1.15), and Deb's spread metric [29] (Equation 1.16). The GD and HV were used to measure the closeness/quality of the non-inferior solutions provided by the different approaches. The OS and the spread were used to measure the diversity present in the non-inferior solutions provided by the different approaches. A lower value for the GD (Equation 1.13) and Deb's spread metric (Equation 1.16) would indicate a better performance value. The GD (Equation 1.13) represents the distance between the obtained non-inferior solutions and the reference solutions, so having a lower value for the GD (Equation 1.13) would indicate that the obtained solutions line up with the reference solutions for the problem. Deb's spread metric (Equation 1.16) compares the distance between adjacent points in the non-inferior solution to the average distance between all adjacent points. So when this metric (Equation 1.16) has a lower value, it signifies that the points along the non-inferior solutions are evenly spread out. A higher value for HV (Equation 1.14) and OS (Equation 1.15) indicates a better performance value. HV (Equation 1.14) is calculated in reference to the nadir point for the problem, so the largest possible hypervolume that could be obtained for the problem is when the obtained solution lines up with the Pareto front for the problem. The OS (Equation 1.15) shows how much of the volume/area is occupied by the obtained non-inferior solutions in reference to the space occupied by the reference solutions (the largest possible OS (Equation 1.15) value that could be calculated for this is 1, which indicates that the obtained solutions occupy the same space that the reference solutions do).

To provide a fair comparison between the different algorithms, the same number of function calls was allocated for each of the test problems. The number of allowable function calls for

different test problems was determined based on the number of decision variables present in the optimization problem. This calculation was 50 times the number of decision variables, so if there are 10 decision variables, 500 function calls would be allowed for the different algorithms. The summary of the characteristics of the different optimization problems is shown in Table 3.1. The reference non-inferior solutions for each of these test problems were calculated using the non-inferior set of running NSGA2 [29] 10 times on the given test problem until the optimizer was able to converge to a set of solutions. The reason for running NSGA2 [29] multiple times is because it is a stochastic optimizer that has uncertainty present in it's calculations so by taking the non-inferior solution of multiple runs, this would provide a better representation for the Pareto front for the given test problem.

Table 3.1: Test Problem Description Summary

Problem Name	No. Objectives	No. Constraints	No. Decision Variables	No. Allowed Function Calls
TNK	2	2	2	100
OSY	2	6	6	300
CTP	2	1	10	500
ZDT9	2	1	20	1000
ZDT9	2	1	40	2000
ZDT9	2	1	100	5000
ZDT9	3	2	30	1500
ZDT9	3	2	60	3000
ZDT9	3	2	150	7500

3.2 Illustrative Example

The TNK test problem [34] (Equation 3.1) was used to show the process and functionality of the proposed approach. The process for the proposed approach is visually depicted in Figures 3.1, 3.2, and 3.3.

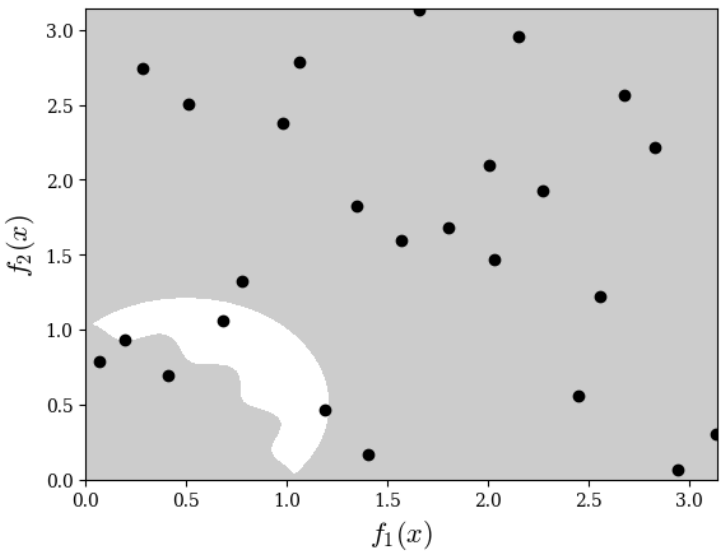

Figure 3.1: Initial Sampling of TNK Test Problem Showing True Representation of Feasible and Infeasible Regions

$$
\begin{aligned}
&minimize f_1(x) = x_1 \\
&minimize f_2(x) = x_2 \\
&s.t. \\
&G_1(x) = x_1^2 + x_2^2 - 1 - 0.1\cos(16\arctan(\frac{x_1}{x_2})) \geq 0 \\
&G_2(x) = (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \leq 0.5
\end{aligned}
\tag{3.1}
$$

The first step of the proposed approach (Algorithm 1, shown in Chapter 2) was performing an initial sampling. The Latin Hypercube [22] was used as the sampling method in this example. A visual of the initial sampling is shown in Figure 3.1. The greyed-out region of Figure 3.1 represents the true infeasible region for the problem, while the white space represents the region where the feasible points are located. The black points are the locations of the points that were sampled using the Latin Hypercube [22].

The SVM constraint surrogate models were trained on the available sample points. These

sampled points are depicted as the black points in Figure 3.2. The trained SVM constraint surrogate models were used to create a boundary between the feasible and infeasible points for each of the constraints. Figure 3.2 illustrates the aggregated predicted feasible and infeasible regions. The white space represents the predicted feasible region, where all the constraints are satisfied, while the gray space indicates the predicted infeasible region, where at least one constraint is violated. After the SVM constraint surrogate models have been trained, they are used to inform the GAN model. The GAN is then used to generate a large sample of possible offspring points that are shown by the hollow gray points in Figure 3.2. From the set of possible offspring points, various characteristics of the points (predicted feasibility, predicted fitness quality, and distance within the design space from previously sampled points) were considered to select the most suitable point (shown by the gray star in Figure 3.2) for evaluation in the TNK problem and update the algorithm with the new information. This process is repeated until the algorithm has used all the available function calls allocated by the user. Figure 3.3 presents the non-inferior feasible solutions obtained from a single run using 100 function calls and another run using 1,000 function calls, illustrating the impact of increasing the number of function calls in terms of diversity of the non-inferior solutions and the closeness of these non-inferior solutions to the Pareto front.

Comparing the two cases (100 function calls and 1000 function calls) in Figure 3.3 demonstrates that increasing the number of allowable function calls enhances the diversity and accuracy of the non-inferior feasible solutions. With more function calls, the proposed approach generates solutions that are more widely distributed and closer to the true Pareto front, improving the overall quality of the optimization results. The increased diversity is shown by the non-inferior solution set for the 1000 function call trial having more points in the upper-left region of Figure 3.3

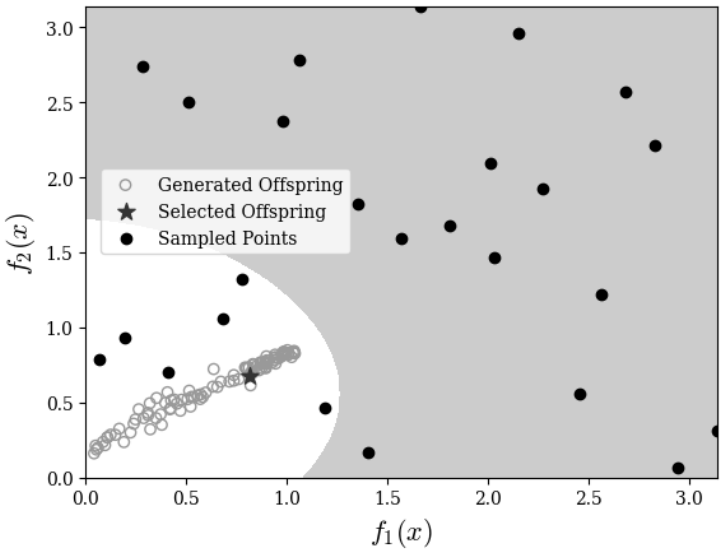

Figure 3.2: Point Generation and Point Selection for the Next Generation based on the Initial Population for the TNK Test Problem

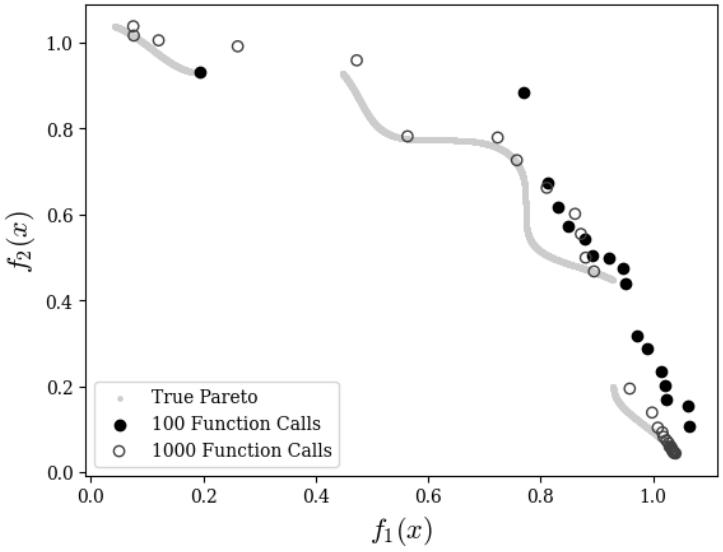

Figure 3.3: TNK Test Problem Non-Inferior Feasible Solutions after Final Generation for a Single 100 and 1000 Function Call Trial

compared to the non-inferior solution set for the 100 function call trial. The increased accuracy of the non-inferior solution set in the 1000 function call trial compared to the 100 function call trial is shown by the number of points that fall on the true Pareto Front for the TNK test problem (Equation 3.1). There is only one point that is directly touching the true Pareto Front for the TNK test problem (Equation 3.1) for the 100 function call trial, whereas there are significantly more points touching the true Pareto Front for the 1000 function call trial.

3.3 Test Problem Formulations

In addition to the TNK test problem (Equation 3.1), shown in section 3.2, the other test problems used to evaluate the different approaches are shown in this section. The formulation for the OSY test problem [37] is shown in Equation 3.2.

$$
\begin{aligned}
minimize f_1(x) &= -(25(x_1-2)^2 + (x_2-2)^2 \\
&\quad + (x_3-1)^2 + (x_4-4)^2 + (x_5-1)^2) \\
minimize f_2(x) &= \sum_{i=1}^{6} x_i^2 \\
s.t.\ G_1(x) &= x_1 + x_2 - 2 \geq 0 \\
G_2(x) &= 6 - x_1 - x_2 \geq 0 \\
G_3(x) &= 2 - x_2 + x_1 \geq 0 \\
G_4(x) &= 2 - x_1 + 3x_2 \geq 0 \\
G_5(x) &= 4 - (x_3-3)^2 - x_4 \geq 0 \\
G_6(x) &= (x_5-3)^2 + x_6 - 4 \geq 0 \\
0 \leq x_1, x_2, x_6 \leq 10,\ 1 \leq x_3, x_5 \leq 5,\ 0 \leq x_4 \leq 6
\end{aligned}
\tag{3.2}
$$

The formulation for the CTP test problem [29] is shown in Equation 3.3.

$$
\begin{aligned}
&minimize f_1(x) = x_1 \\
&minimize f_2(x) = g(x)(1 - \sqrt{\frac{f_1(x)}{g(x)}}) \\
&s.t.\ g(x) = |1 + (\sum_{i=2}^{10} x_i)^{0.25}| \\
&G(x) = \cos(\theta)(f_2(x) - e) - \sin(\theta) f_1(x) \geq a|\sin(b\pi(\sin(\theta)(f_2(x) - e) + \cos(\theta) f_1(x))^c)|^d \\
&\theta = -0.2\pi, a = 0.2, b = 10, c = 1, d = 6, e = 10 \\
&0 \leq x_1 \leq 1 \\
&-5 \leq x_i \leq 5, \quad i = 2, ..., 10
\end{aligned}
\tag{3.3}
$$

The formulation for the ZDT9 test problem [38] is shown in Equation 3.4. The number of decision variables (x_i) was recommended to be a multiple of 10 times the number of objectives (M). The number of constraints is set to be one less than the number of objectives. For this reason, the ZDT9 test problem was configured with 20, 40, and 100 decision variables in the two-objective formulation to evaluate the algorithm's scalability and performance across different levels of complexity. The ZDT9 had a 30, 60, and 150 decision variable configuration for the three-objective formulation.

$$
\begin{aligned}
minimize f_j(x) &= \sum_{i=\frac{(j-1)n}{M}}^{j\frac{n}{M}} x_j^{0.1}, j = 1, 2, ..., M \\
s.t.\ g_j(x) &= f_M^2(x) + f_j^2(x) - 1 \geq 0, \\
for j &= 1, 2, ..., (M-1) \\
0 \leq x_i &\leq 1, for\ \ i = 1, 2, ..., n
\end{aligned}
\tag{3.4}
$$

3.4 Test Problem Results and Discussion

The results for the TNK test problem (Equation 3.1) shown in Figure 3.4 shows that the proposed approach is able to outperform GMOEA and Forrester method. This is done by showing that the majority of the points in the worst case can dominate the majority of the points from the best case of GMOEA, and the Forrester method. The best case of the proposed approach can cover almost the entire reference non-inferior solutions for the TNK test problem (Equation 3.1), with many points touching the reference non-inferior solutions.

For the OSY test problem (Equation 3.2) shown in Figure 3.5, it can be seen that none of the approaches (proposed approach, GMOEA, or Forrester method) were able to capture the leftmost region of the objective space. The likely cause for this issue is that none of the approaches performed sufficient exploration of the design space and narrowed in on the decision variable values that resulted in points on the right side of Figure 3.5. This could be solved if more function calls were used, and a more diverse initial sampling could occur during the initial generation. The optimizer would then have more function calls available for sampling points from a more diverse region of the design space. Despite this shortcoming, Figure 3.5 shows that the proposed approach is able to outperform GMOEA and the Forrester method for the OSY test problem

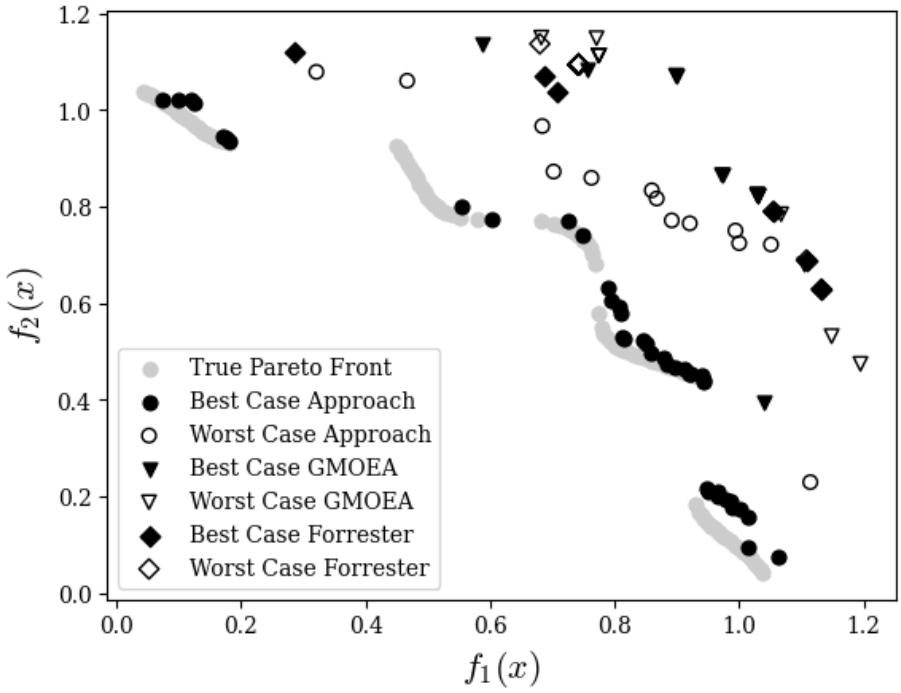

Figure 3.4: Best/Worst Case Representation for Proposed Approach, GMOEA, and Forrester method on TNK Test Problem

(Equation 3.2). This is evident by the worst case of the proposed approach being closer to the reference non-inferior solutions compared to the best case of GMOEA or the best case of the Forrester method.

Figure 3.6 shows that visually the proposed approach, GMOEA, and Forrester method are able to generate points that fall within a similar region in the objective space for the CTP test problem (Equation 3.3). However, it is shown that the proposed approach can generate points that are close to the reference non-inferior solutions for the full range of values present in Figure 3.6.

Figure 3.7 shows that the approach that is able to generate solutions closest to the reference non-inferior solutions is the proposed approach for the ZDT9 test problem with 20 decision variables and two objectives (Equation 3.4). These points also contain a diverse range of values

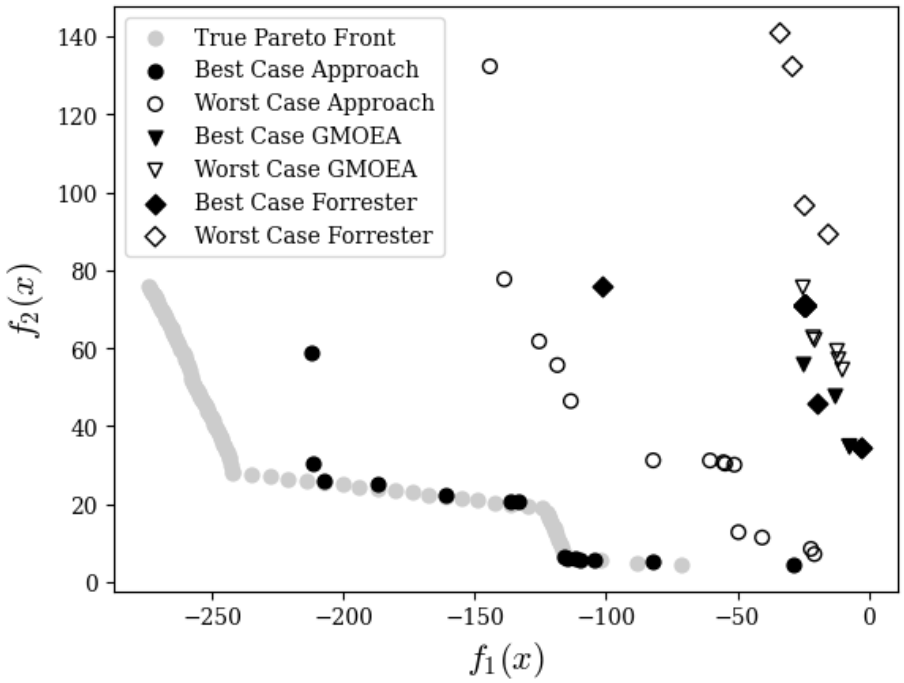

Figure 3.5: Best/Worst Case Representation for Proposed Approach, GMOEA, and Forrester method on OSY Test Problem

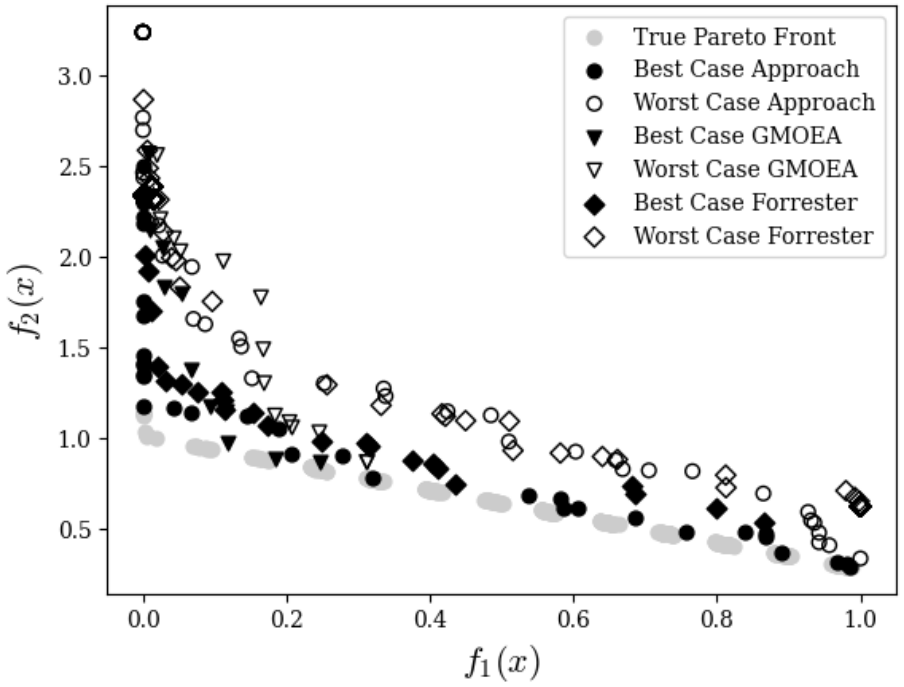

Figure 3.6: Best/Worst Case Representation for Proposed Approach, GMOEA, and Forrester method on CTP Test Problem

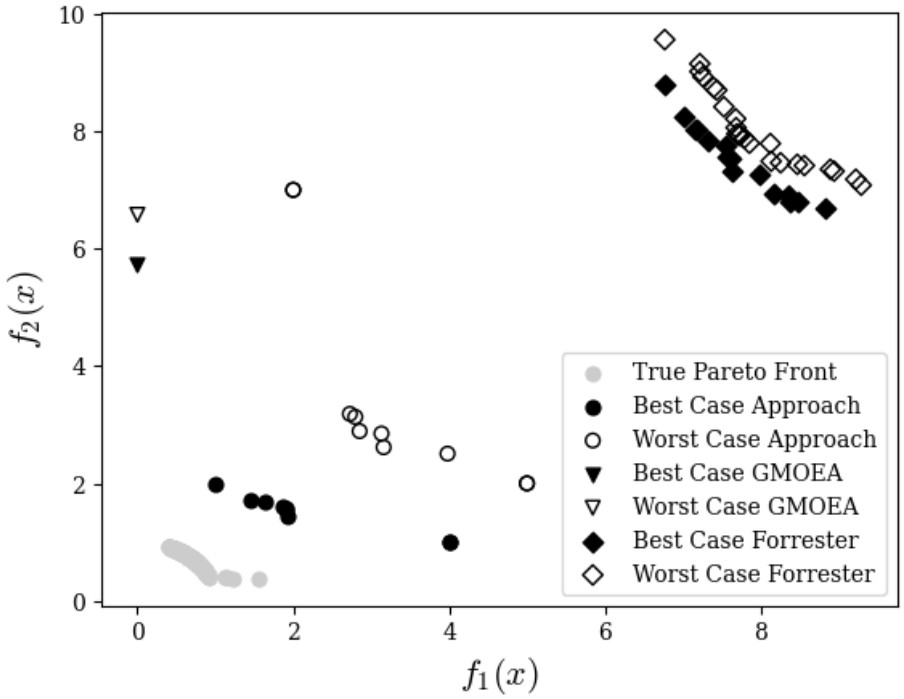

Figure 3.7: Best/Worst Case Representation for Proposed Approach, GMOEA, and Forrester method on ZDT9 Test Problem using 20 Decision Variables

in both objectives. The proposed approach can significantly outperform the Forrester method since the worst case of the proposed approach is able to dominate all the points in the best case of the Forrester method. All the points in the best case of the proposed approach are closer to the reference non-inferior solutions compared to the points generated by GMOEA. It is also shown visually that the proposed approach has more diverse points compared to GMOEA, as GMOEA seems to only generate a single point for the best/worst case.

Figure 3.8 shows that the proposed approach is able to significantly outperform GMOEA and Forrester method for the ZDT9 test problem (Equation 3.4) with 40 decision variables and two objectives. The proposed approach can significantly outperform GMOEA and Forrester method, as the proposed approach is significantly closer to the reference non-inferior solutions compared to GMOEA or Forrester method.

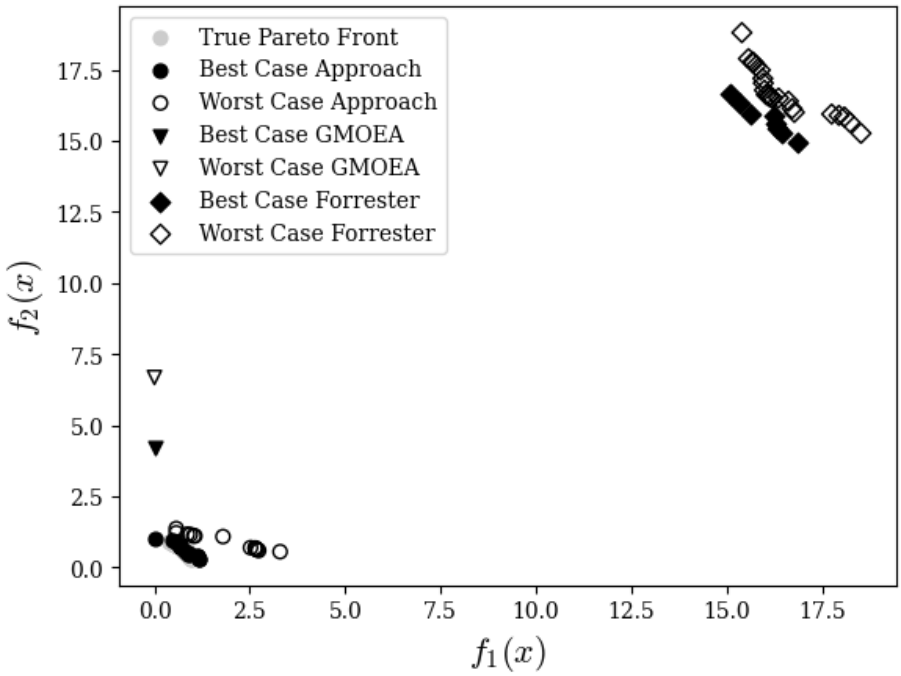

Figure 3.8: Best/Worst Case Representation for Proposed Approach, GMOEA, and Forrester method on ZDT9 Test Problem using 40 Decision Variables

Figure 3.9 shows that the proposed approach is able to outperform GMOEA for the ZDT9 test problem (Equation 3.4) with 100 decision variables and 2 objectives. GMOEA is shown to have little diversity as it seems to only have a single point for the best case and a single point for the worst case solutions, whereas the proposed approach can generate multiple non-inferior points in both the best case and worst case conditions. The proposed approach can produce non-inferior solutions closer to the reference non-inferior solutions compared to GMOEA. Forrester method could not be tested for the ZDT9 test problem (Equation 3.4) with 100 decision variables and 2 objectives because the computational time to evaluate each test problem for a single run exceeded 4 hours. Due to this computational expense, the Forrester method could not be evaluated for any of the test problems that had 60 or more decision variables.

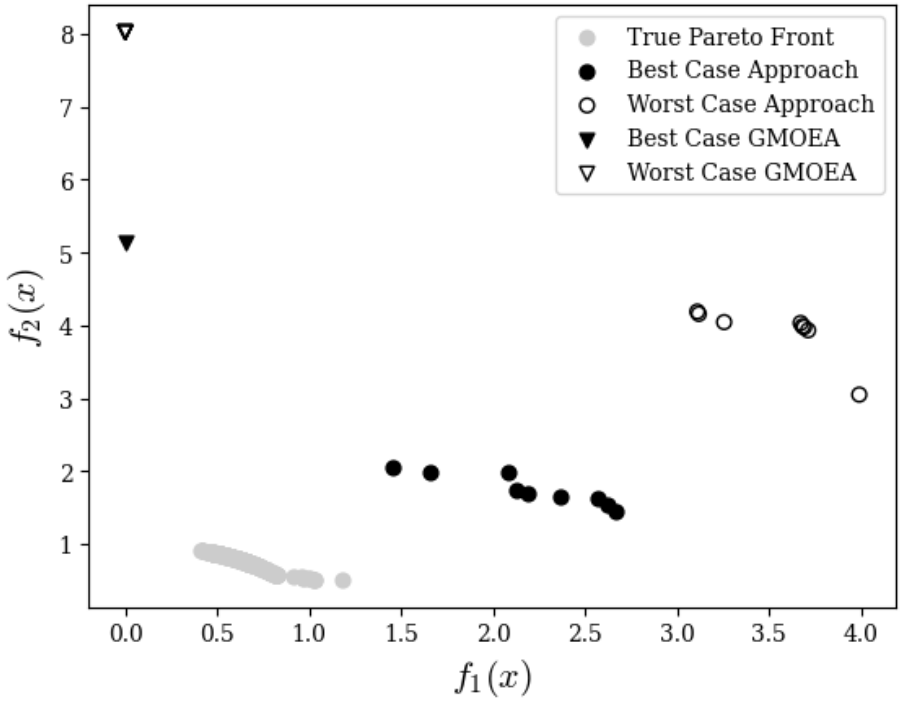

Figure 3.9: Best/Worst Case Representation for Proposed Approach, GMOEA, and Forrester method on ZDT9 Test Problem using 100 Decision Variables

3.4.1 Closeness Evaluation

The quality metrics that evaluate the closeness of the non-inferior solutions for a given approach are shown in Table 3.2 (GD) and Table 3.3 (HV). Based on the GD (Table 3.2, Equation 1.13), the smaller the GD value, the better the performance. The smaller the GD value, the closer the Euclidean distance between the obtained non-inferior solutions and the reference non-inferior solutions for the problem. The proposed approach performed the best with statistical significance for 8 of the 9 test problems. The differences in performance between the approaches were shown to have statistical significance based on the ANOVA test [42]. This was proven by the calculated p-value being less than 0.05.

Table 3.2: Generational Distance Quality Metrics for Test Problems (values are reported as mean (standard deviation), with the best performing values in bold, smaller values indicates better performance)

Test Problem	GMOEA	Forrester	Proposed Approach
TNK	0.31 (0.046)	0.29 (0.025)	**0.079** (**0.048**)
OSY	76 (4.7)	103 (24)	**18** (**14**)
CTP	**0.42** (**0.21**)	0.73 (0.12)	1.0 (0.23)
ZDT9 20-Var	5.6 (0.18)	11 (0.037)	**2.5** (**0.68**)
ZDT9 40-Var	5.7 (0.75)	24 (0.033)	**0.36** (**0.19**)
ZDT9 100-Var	6.8 (0.64)	–	**2.6** (**0.93**)
ZDT9 30-Var	11 (0.19)	8.0 (0.15)	**1.1** (**1.0**)
ZDT9 60-Var	19.7 (0.62)	–	**5.6** (**0.50**)
ZDT9 150-Var	43 (1.6)	–	**21** (**0.65**)

Based on the HV (Table 3.3, Equation 1.14), the larger HV value indicates a better performance. This is a measure of closeness because the optimizer is set up to minimize all objectives. Therefore, the ideal point for the multi-objective optimization problem would be as close as possible to negative infinity, as allowed by the problem, for all objectives. Since the calculation is in terms of the nadir point of the problem, the larger the HV, the more space is occupied between the nadir point and the obtained non-inferior set of solutions, indicating that the obtained non-inferior set of solutions is closer to the ideal point.

The proposed approach outperformed the other methods with statistical significance on 7 out of the 9 test problems. These seven test problems showed significant improvement, with p-values less than 0.05 as determined by the ANOVA test [42] indicating that the observed

performance differences were not due to random chance. The ZDT9 20-variable configuration and the ZDT9 100-variable configuration did not have a statistical difference for the HV calculation. This is because the statistical significance in the difference between the HV quality metric for these two problems is that the final solution set provided by all three approaches yielded values that were inferior in comparison to the reference point used for the calculation of HV, which resulted in a score of zero for these test problems.

Table 3.3: Hypervolume Quality Metrics for Test Problems (values are reported as mean (standard deviation), with the best performing values in bold, larger value indicates better performance)

Test Problem	GMOEA	Forrester	Proposed Approach
TNK	0.054 (0.029)	0.062 (0.016)	**0.33** (**0.051**)
OSY	0.69 (2.1)	0.0 (0.0)	**7400** (**17**)
CTP	0.25 (0.091)	0.38 (0.030)	**0.47** (**0.057**)
ZDT9 20-Var	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
ZDT9 40-Var	0.0 (0.0)	0.0 (0.0)	**0.032** (**0.022**)
ZDT9 100-Var	0.0 (0.0)	–	0.0 (0.0)
ZDT9 30-Var	0.0 (0.0)	0.0 (0.0)	**8.9** (**10.0**)
ZDT9 60-Var	0.0 (0.0)	–	**2200** (**101.0**)
ZDT9 150-Var	0.0 (0.0)	–	**30100** (**1200**)

3.4.2 Diversity Evaluation

The quality metrics that measured the diversity of the non-inferior solutions for a given approach are shown in Table 3.4 (OS) and Table 3.5 (Spread). Based on the OS (Table 3.4,

Equation 1.15), the proposed approach was able to perform the best for 6 of the 9 test problems. The larger OS value indicates a better performance. The reason for this OS is a measure of how much of the overall rectangular or cubic space occupied by the reference non-inferior solution set overlaps with the overall rectangular or cubic space occupied by the obtained non-inferior solutions.

All the test problems had statistical significance (p-value less than 0.05) based on the ANOVA test [42] except for the ZDT9 20-variable, ZDT9 100-variable, and the ZDT9 150-variable test problem configurations. The lack of statistical significance for these problems in the ANOVA test [42] can be attributed to the fact that the non-inferior solutions generated by the different approaches produced values that were worse than the reference point used to calculate the overall spread. As a result, the calculated values for the OS for the different approaches for the ZDT9 20-variable, ZDT9 100-variable, and ZDT9 150-variable test problems were zero/near zero.

Table 3.4: Overall Spread Quality Metrics for Test Problems (values are reported as mean (standard deviation), with the best performing values in bold, larger value indicates better performance)

Test Problem	GMOEA	Forrester	Proposed Approach
TNK	0.0024 (0.0074)	0.014 (0.043)	**0.20** (**0.20**)
OSY	0.0 (0.0)	$5.4e-6$ $(1.1e-5)$	**0.23** (**0.11**)
CTP	0.044 (0.091)	0.62 (0.054)	**0.76** (**0.17**)
ZDT9 20-Var	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
ZDT9 40-Var	0.0 (0.0)	0.0 (0.0)	**0.29** (**0.25**)
ZDT9 100-Var	0.0 (0.0)	–	0.0 (0.0)
ZDT9 30-Var	0.0 (0.0)	0.0 (0.0)	**0.0077** (**0.0094**)
ZDT9 60-Var	0.0 (0.0)	–	**0.0016** (**0.0021**)
ZDT9 150-Var	0.0 (0.0)	–	$\mathbf{8.9e-7}$ $(\mathbf{2.2e-6})$

Based on the spread metric (Table 3.5, Equation 1.16), the proposed approach was able to outperform the others in 4 of the 9 test problems. The spread metric is a measure of how uniformly distributed the obtained non-inferior solution set is. The smaller the value, the closer the distance between adjacent points on the non-inferior solution set is to the average distance between adjacent points. So the smaller the spread metric value, the more uniformly distributed the points in the obtained non-inferior solution set are, and thus are better performing.

All the test problems had statistical significance (p-value less than 0.05) based on the ANOVA test [42] in terms of the spread metric (Equation 1.16). A possible reason for the relatively lower performance of the proposed approach is due to the GAN. The GAN tended to generate offspring points concentrated in a specific region of the design space, limiting its

ability to explore the entire search space. This behavior led to insufficient exploration during the later iterations of the optimization process, causing the algorithm to get stuck in local optima and hindering its overall performance in finding diverse solutions.

Table 3.5: Spread Quality Metrics for Test Problems (values are reported as mean (standard deviation), with the best performing values in bold, smaller value indicates better performance)

Test Problem	GMOEA	Forrester	Proposed Approach
TNK	1.0 (0.042)	1.1 (0.11)	**0.93** (**0.071**)
OSY	1.0 (0.0)	1.0 (0.095)	**0.81** (**0.11**)
CTP	**0.89** (**0.068**)	1.5 (0.011)	1.3 (0.28)
ZDT9 20-Var	1.1 (0.041)	**0.65** (0.014)	0.97 (0.035)
ZDT9 40-Var	1.3 (0.070)	**0.63** (**0.013**)	0.86 (0.16)
ZDT9 100-Var	1.2 (0.082)	–	**0.97** (**0.022**)
ZDT9 30-Var	1.0 (0.18)	**0.51** (**0.024**)	0.80 (0.23)
ZDT9 60-Var	0.96 (0.18)	–	**0.94** (**0.062**)
ZDT9 150-Var	**0.85** (**0.15**)	–	1.0 (0.012)

Based on these results, the proposed approach is shown to have a statistically significant advantage in performance on these quality metrics in comparison to GMOEA and Forrester method, as explained in the analysis of each of the test problems earlier. Looking at the trend shown throughout all the results for the ZDT9 test problems (Equation 3.4) shows that the proposed approach is able to maintain the increased performance as the number of decision variables/number of objectives and constraints scale over GMOEA and Forrester method. The proposed approach was able to outperform GMOEA and Forrester method in regards to GD

(Equation 1.13) and HV (Equation 1.14) for more test problems compared to OS (Equation 1.15) and spread (Equation 1.16). However, for the OS (Equation 1.15) and spread (Equation 1.16), the proposed approach was able to perform the best out of the three approaches tested for more test problems than either GMOEA or Forrester method.

3.5 Summary

This chapter provided a comprehensive performance evaluation of the proposed approach against GMOEA and the Forrester method. Visual and quantitative comparisons through nine test problems from established literature were used to compare the performance of the different approaches. From these test problems, it was shown that the proposed approach was able to outperform GMOEA and the Forrester method consistently in terms of GD (Equation 1.13) and HV (Equation 1.14). This showed that the proposed approach has an advantage over GMOEA and the Forrester method in terms of closeness. In addition to this, the proposed approach was able to outperform GMOEA and the Forrester method for many of the test problems in terms of diversity. This was shown by OS (Equation 1.15) and spread (Equation 1.16) but did not have the same level of advantage that was seen in the GD (Equation 1.13) and HV (Equation 1.14). Despite the limitations in regards to the diversity for a few of the test problems, which was likely attributed to the behavior of the GAN, the proposed approach demonstrated a scalable and competitive performance in the different tested scenarios. These results validate the effectiveness of the proposed approach.

In the next chapter, the proposed approach is applied to a complex engineering optimization problem. This problem is for the optimization of the throttle control profile for a USV. This

shows the application of the proposed approach for a complex black-box optimization problem that could be seen in the real world taking it a step further than what was demonstrated in these test problems.

Chapter 4: Unmanned Surface Vessel Case Study

As the development of unmanned systems continues, such as autonomous marine vessels, the optimization of the control for these vessels through uncertain conditions becomes increasingly important. This chapter presents a case study that applies the proposed approach (described in Chapter 2) to a real-world engineering problem, where the engine's throttle profile of a USV was determined when it undertakes a 200-hour mission from San Diego to Hawaii. The objectives for this problem are to maximize the expected system reliability and the expected speed of the USV. Realistic operating conditions are added so that the optimizer would need to make considerations for additional factors that could occur during the mission. These realistic operating conditions would include ambient air temperature, ambient air pressure, seawater temperature, and seawater current speed. These are all factors from the environment that would be expected to affect the performance of a USV. More information about the specifics of these factors will be discussed in Section 4.1.

This chapter demonstrates how a complex, physics-based model of a USV's engine cooling and control system can be integrated into an optimization framework. It shows the computational limitations of traditional optimization methods due to the computational expense of the physics-based simulation and how that motivates the use of online surrogate-based multi-objective design optimizers, such as the proposed approach.

This chapter starts in Section 4.1 by introducing the structure and function of the USV's engine cooling and control system (ECCS), and the environmental uncertainties that affect the synthetic sensor data provided by the physics-based simulation model. Section 4.2 describes the structure for the physics-based simulation model, how synthetic sensor data was generated, and how degradation/damage and reliability were incorporated into the simulation model. Section 4.3 discusses the development of an offline surrogate model that could be used in place of running the physics-based simulation model along with an unconstrained bi-objective optimization problem this was applied to. Section 4.4 presents a constrained bi-objective optimization problem that expands on the previous work done with the unconstrained optimization problem. This constrained bi-objective optimization problem was solved using the different online surrogate-based multi-objective design optimizers that were shown in Chapter 3 (GMOEA [17], the Forrester method [11], and the proposed approach). Section 4.5 provides and compares the results for how the three approaches performed on this close-to real-world application through the use of the closeness and diversity quality metrics used in Chapter 3.

4.1 Engine Cooling and Control System Description

This case study aimed to identify the optimal throttle values required to establish the operating profile for effectively controlling an unmanned surface vessel (USV). The operating profile consists of five throttle values, each marking the percentage of total engine load the USV is exerting for a set duration (for the 200-hour mission, each throttle value would be held for 40-hour increments). The USV is assumed to undertake a trip between San Diego and Hawaii with a given mission time of 200 hours. This case study considers uncertainty present in the

environment, such as the ambient temperature, ambient pressure, seawater temperature, and sea current speed.

A simulation model was used to collect synthetic sensor data that a person would expect to be present on the USV. Due to the complexity of creating a digital twin for a complete USV, specific subsystems were chosen to simulate instead of simulating all aspects of a complete USV. The subsystems chosen include the control of the USV via the engine, and the expected cooling requirements that this engine would need. Together these subsystems create the engine cooling and control system. The engine is responsible for generating the propulsion the USV needs to operate, and the cooling ensures that the engine can perform its duties without significant damage.

Data was collected on the USV under various environmental load and throttle profiles. The environmental load involved generating a random temperature and pressure profile for the environment based on realistic conditions of the environment between San Diego and Hawaii. The upper and lower bounds for these environmental loads were assumed to change linearly with time. The environmental load was created by sampling a uniform random distribution between these bounds. The environmental conditions, shown in Table 4.1, were based on published data found in [43] and [44]. Each of the throttle stages that make up the overall throttle profile for the mission is represented with values between 0 (idle operation of the engine) and 1 (maximum engine load).

The engine is designed to generate more heat at higher loads. The heat generation needs to be siphoned off to the environment so that it does not damage the system. This is done through four cooling subsystems (air, freshwater, seawater, and lube oil). The air, lube oil, and freshwater subsystems pull heat directly from the engine. The lube oil transfers heat to the freshwater subsystem. The air and freshwater subsystem transfers heat to the seawater subsystem, where

Table 4.1: Environment profile uncertainties.

Environment	Lower bound		Upper bound	
	Start	Finish	Start	Finish
Air Temperature (°C)	10	20	22	34
Seawater Temperature (°C)	19	22	25	28
Air pressure (MPa)	0.1006	0.1006	0.102	0.102
Sea current (m/s)	0.5	0.5	1	1

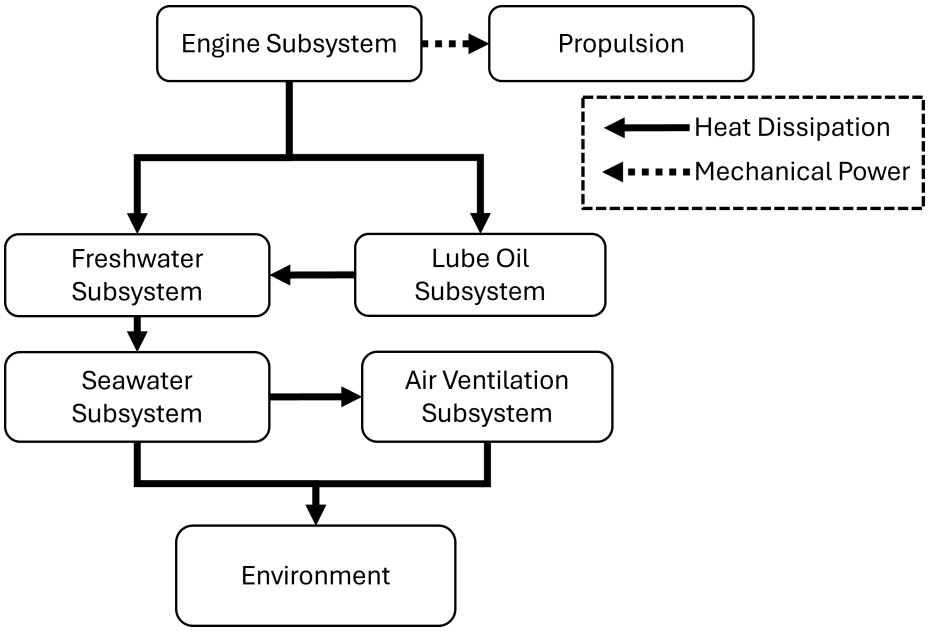

Figure 4.1: Engine Cooling and Control System

the heat is then released into the environment. This process is shown in figure 4.1.

Simulating the cooling system to siphon off heat provides a measurable indicator for the health of the system and subsystems. This means that if the system and subsystems are in a healthy operational state, then the temperatures should be relatively lower compared to when they are in an unhealthy operational state. When the system and its subsystems are in an unhealthy operational state, the rate at which heat is siphoned off the USV and dissipated into the environment decreases. This reduction leads to heat buildup in certain subsystems and components, which can be detected through elevated temperature readings.

4.2 Model Description and Data Generation

The physics-based simulation model was developed using MATLAB's Simulink and Simscape library [45]. This software was used due to its extensive physics-based simulation components. This allowed the simulation to take into account the loads placed onto the system and subsystems due to the environment and loads caused by the operation of the USV. This was controlled using a Python script to generate, control, and organize the data that is fed into and received from the simulation. The Python script also handled the integration of other components of the model such as interactions with a DBN to calculate the system's reliability. Temperature and pressure sensors were placed throughout the ECCS system to represent the possible sensors that could be collected from the USV.

The engine was based on a six-cylinder marine engine design. The heat generated by the engine was calculated based on the amount of mechanical power the engine was outputting at any given moment during the simulation. This method was used to calculate the heat because if the system requires more mechanical power, the engine must generate more power, which consequently results in the production of more heat as a by-product. The amount of heat generated by the engine considers environmental impacts and the efficiency of the engine. This engine was set to have a nominal heat-to-mechanical power ratio of 51.5%:48.5%; this ratio was based on work done in [46]. This was modified using the adjustment factor of β which would account for how much environmental and internal reliability of the engine would impact the heat generation (Equation 4.1).

$$
\begin{aligned}
k &= (\frac{P_0}{P_{ref}})^m(\frac{T_{ref}}{T_0})^n(\frac{T_{s1,ref}}{T_{sw}})^s \\
\alpha &= k - 0.7(1-k)(\frac{1}{\eta_{eng}} - 1) \\
\beta &= \frac{k}{\alpha}
\end{aligned}
\tag{4.1}
$$

For equation 4.1, k represents the adjustment factor of the engine from environmental conditions, α represents the mechanical power adjustment factor, and β represents the adjustment factor for the specific fuel consumption. β can be used as an adjustment factor for the amount of heat the engine generates since more heat is generated as more fuel is consumed. P_0 is the ambient air pressure, P_{ref} is the reference air pressure, T_0 is the air temperature, T_{ref} is the reference air temperature, T_{sw} is the seawater temperature, $T_{sw,ref}$ is the reference seawater temperature, m, n, and s are curve fitting importance parameters used to tune the readings for a specific diesel engine, and η_{eng} is the mechanical efficiency of the engine. The reference temperatures and pressures were obtained from [47], and [48]. For the reliability of a subsystem to create a physical response in the synthetic sensor data, the subsystem degradation needed to be incorporated into the physics-based simulation model. Changing the efficiency of the subsystem based on the healthy state, and how much degradation the subsystem has experienced, would create a measurable change in the synthetic sensor data generated by the physics-based simulation model. The efficiency was modeled using equation 4.2.

$$
\eta = \frac{p_{in} - p_{leak}}{p_{in}} \tag{4.2}
$$

where p_{in} represents the input power into the component, p_{leak} represents the amount of power lost by the component, and η represents the efficiency of the component. As the component

degrades the amount of leakage present increases. The ECCS is assumed to follow an exponential growth model (Equation 4.3), that is offset to represent the damage that would be present from previous use/manufacturing defects. In equation 4.3, u represents the uncertain parameters created by the environment and manufacturing defects, x represents the given throttle control profile, t represents the time since the start of the mission, λ represents the calculated failure rate at a given moment given the throttle control profile and the uncertain parameters, and R represents the reliability of the subsystem/component.

$$R(u, x, t) = e^{-\lambda(u,x)t} \tag{4.3}$$

The failure rate calculation for each subsystem was based on information from reliability handbooks ([49], [50]). The effect of decreased reliability within the simulation is shown by increasing the loss component (p_{leak}) in the efficiency equation (Equation 4.2). The growth of the loss component followed an exponential growth model (Equation 4.4) that matches the assumed reliability calculation (Equation 4.3), with an additional component of p_{offset} representing the offset factor so that there is an initial non-zero loss value. This offset amount is used to simulate different manufacturing defects that could appear in the components.

$$p_{leak} = (p_{in} - p_{offset})(1 - e^{-\lambda(T,Q,N)t}) + p_{offset} \tag{4.4}$$

The loss calculation (Equation 4.4) is introduced into the efficiency equation (Equation 4.2), and this results in an equation for calculating the efficiency of the component, as shown in equation 4.5. This was used to model the efficiency of the different subsystems.

$$\eta = (1 - p_{offset})e^{-\lambda_{pump}(T,Q,N)t} \tag{4.5}$$

As the engine is more complex than a pump, the engine's efficiency was augmented using k from equation 4.1 as this would also consider the negative impacts that the environment would have on the engine's efficiency. This results in equation 4.6 as shown below.

$$\eta_{eng} = (1 - p_{offset})e^{\frac{-\lambda_{eng}t}{k}} \tag{4.6}$$

Once the synthetic sensor data was generated, a CNN was used to predict the reliability of the subsystems. The predicted reliability of the subsystems was then fed into a DBN to calculate the overall system reliability.

4.3 Offline Surrogate Model Development

Due to the complexity of the simulation, an offline surrogate model was developed. Without an offline surrogate model, a single iteration of an optimizer would take 45 minutes to run on a desktop computer with an Intel Core i9-12900KF processor. This computational time proves that the model has a significant computational expense that would be impractical to use for optimization. Instead, a Monte Carlo simulation method was used to generate random input samples for the simulation to evaluate the construction of an offline surrogate model. The input samples included randomly generated environmental profiles, initial health states for the subsystems, and throttle control profiles for the engine.

The surrogate mode was developed using a Kriging model [11]. This was done using

Python's Scikit-Learn API [51]. The input data for the offline surrogate model included the time-series data containing the throttle profile, initial health states of the subsystems, and environmental loads. This allowed the surrogate model to have access to the same input information that the physics-based simulation model had. To reduce the dimensionality of the input data, principal component analysis (PCA) was performed. The reduced dimensionality was able to account for 99% of the explained variance. This reduced dataset was used to train the Kriging model [11]. This model has an accuracy of over 99% for predicting the speed and the system reliability of the USV. The details for the training and accuracy results of this offline surrogate model are shown in [2].

This offline surrogate model was used to solve the bi-objective optimization problem shown in Equation 4.7.

$$\begin{aligned} &maximize f_1(x) = E_{speed}(x) \\ &maximize f_2(x) = E_{R_{sys}}(x) \\ &s.t.\ 0 \leq x_i \leq 1\ for\ i = 1, ..., 5 \end{aligned} \tag{4.7}$$

This bi-objective optimization problem (Equation 4.7) contained the objectives of maximizing the expected speed and expected system reliability for the USV during the 200-hour mission [2]. The result of this is shown in Figure 4.2. This bi-objective optimization problem (Equation 4.7) was solved using NSGA2 [29] and a Bayesian Optimizer [52]. Since there is uncertainty in the environment, this was handled by generating 100 different environmental conditions and using the expected speed and expected system reliability across all 100 different environmental conditions for the objective calculations.

Multiple multi-objective optimizers were used to validate the non-inferior solution set for

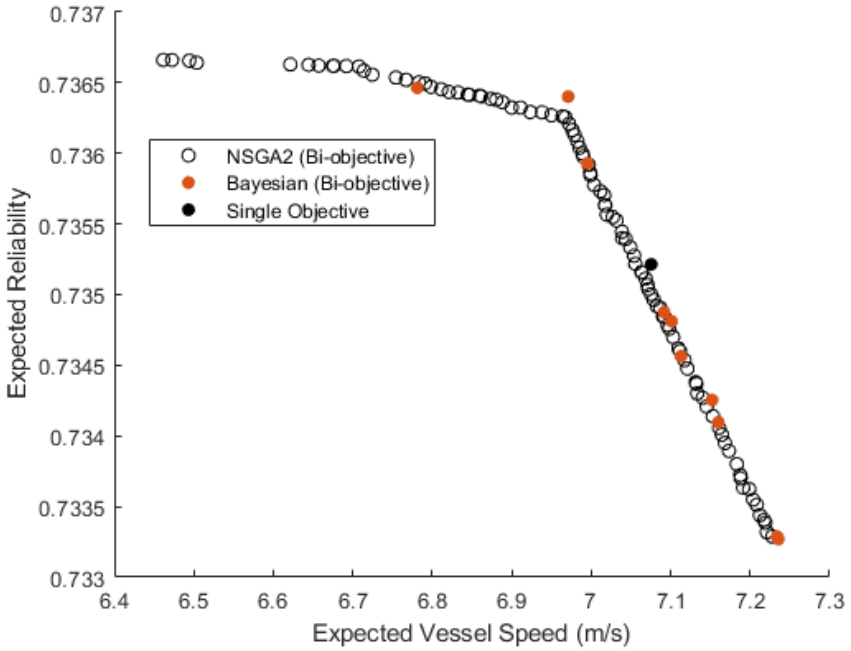

Figure 4.2: Bi-Objective Optimization Non-Inferior Solution Set for Offline Surrogate Model Case Study

the bi-objective optimization problem found by the optimizers to make sure they agreed with each other. This shows two distinct relatively linear regions that comprise the non-inferior solutions for this optimization formulation (Equation 4.7). These two regions in the non-inferior solution set show that when the expected speed of the USV is set to less than 7 m/s the expected system reliability does not change as much compared to when the expected speed of the USV is greater than 7 m/s. When looking at the change that occurs at this point in terms of the decision variables, it is seen that the decision variables associated with an expected speed value greater than 7 m/s in the non-inferior solution set only vary the throttle value at the end of the mission (x_5). All other decision variable values (x_1, x_2, x_3, x_4) had throttle values set to 1 (100%, maximum engine load). However, decision variables associated with an expected speed value less than 7 m/s in the non-inferior solution set vary multiple decision variable values (x_1, x_2, x_3, x_4) while having the throttle value at the end of the mission (x_5) set to the minimum possible engine

load (with a value of 0). As the expected speed reduces from 7 m/s, the throttle values decrease in reverse order (x_4 is set to minimum engine load before decreasing the throttle value of x_3) and this pattern seems to continue.

The above results suggest that the throttle/decision variable values closer to the end of the mission have a more critical impact on the expected system reliability compared to the throttle/decision variable values at the beginning of the mission. Logically this could be explained by the USV having its largest system reliability value at the beginning of the mission, which then decreases as the mission progresses. So there is a large amount of slack available between the expected system reliability at that time and the expected system reliability bound. This decreases as you get to the end of the mission, making the throttle/decision variable value more important. The reliability is assumed to follow an exponential model. Therefore, when the reliability is lower (such as at the end of the mission) the rate at which the damage occurs is significantly higher leading to a more significant impact on the system.

4.4 Online Surrogate Optimization Implementation

Since the building of an offline surrogate model then performing a multi-objective optimization proved to be computationally expensive because it required diverse sampling of the full design space for the problem. This inspired the need to create an online surrogate multi-objective design optimizer. This would sample the regions of the design space where there is a high likelihood of finding non-inferior points and not need to have a diverse sampling of the entire design space. This reduces the computational expense since it is assumed that sampling in the locations where non-inferior points will be will require fewer function calls than performing a diverse sampling

of the entire design space to build an accurate offline surrogate model. In addition to this, other considerations were made to the case study problem (Equation 4.7) to make it more complex by adding constraints. The proposed online surrogate multi-objective design optimizer was used to solve a modified optimization problem shown in Equation 4.8. The reason that the optimization problem from Section 4.3 was modified to create the one shown here (Equation 4.8) is to put in constraints that would be expected to be in place during the actual mission. This would also make it so that the solution to the optimization problem becomes less trivial since there are now solutions that would be infeasible.

$$
\begin{aligned}
maximize f_1(x) &= E_{speed}(x) \\
maximize f_2(x) &= E_{R_{sys}}(x) \\
s.t. G_1(x) &= x_1 \leq 0.75 \\
G_2(x) &= x_5 \leq 0.75 \\
G_3(x) &= -f_1(x) \leq -6.75 \\
0 \leq x_i \leq 1 \; for \; i &= 1, ..., 5
\end{aligned}
\tag{4.8}
$$

In this problem formulation (Equation 4.8), the objectives are to maximize the expected speed of the USV (shown in $f_1(x)$) and maximize the expected system reliability (shown in $f_2(x)$ of the USV. Since there is uncertainty in the environmental conditions, this was handled by generating 100 different uncertain environmental profiles for the USV to travel through, and the expected values of speed and system reliability were used for the calculation of $f_1(x)$ and $f_2(x)$. The constraint $G_1(x)$ is used to limit the maximum engine load percentage during the first stage of the mission. The second constraint $G_2(x)$ is used to limit the maximum engine load percentage

during the last stage of the mission. Both these constraints ($G_1(x)$ and $G_2(x)$) are used to enforce realistic real-world constraints onto the optimization problem. At the beginning and the end of the mission, the USV is likely near/approaching a port where there would be other ships present. In these areas, it would not be safe for the ship to travel full throttle. An additional constraint of having the USV travel faster than $6.75\ m/s$ is used to ensure that the USV travels to the target destination quickly.

The three approaches that were used for the test problems in Chapter 3 (GMOEA, Forrester method, and proposed approach) were used to evaluate this case study (Equation 4.8). This would show how these approaches perform on a practical engineering problem. GMOEA, the Forrester method, and the proposed approach were allowed a total of 250 function calls. All three approaches were given the same number of function calls so that the results for the quality metrics on the non-inferior solution set are comparable.

4.5 Results and Discussion

The quality metrics that were described in Chapter 1 and used to evaluate the test problems in Chapter 3 were used to evaluate the different approaches. Table 4.2 shows the performance of GMOEA, the Forrester method, and the proposed approach for the different quality metrics in the case study. The non-inferior solutions for the best and worst case results of each of the different approaches are shown in Figure 4.3 when they are used to optimize the case study problem (Equation 4.8).

Figure 4.3 shows that all three approaches can find results close to the reference non-inferior set of solutions. The reference non-inferior set of solutions for this problem was obtained

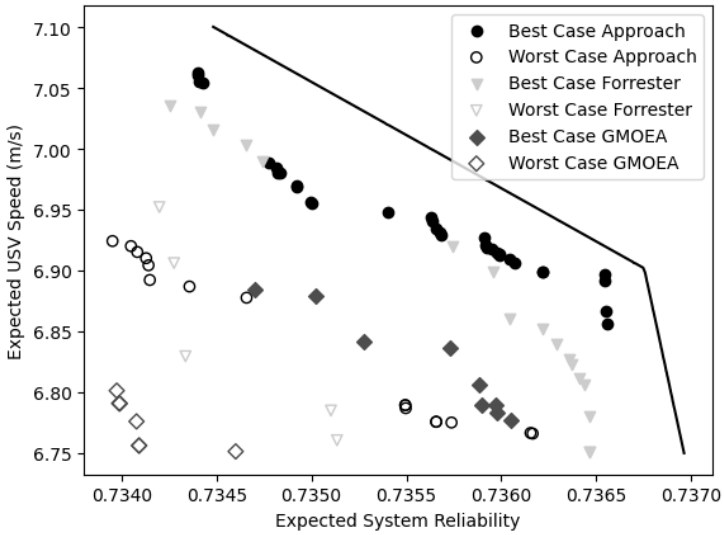

Figure 4.3: Best and Worst Case Representations of Proposed Approach, GMOEA, and Forrester for USV Case Study

using the same process applied to the test problems (Chapter 3). This involved generating the non-inferior set of solutions by running the NSGA2 [29] algorithm 10 times on the case study problem (Equation 4.8) until the NSGA2 [29] optimizer converged. The reference set of solutions is represented as the solid line in figure 4.3. Out of the three approaches tested, the proposed approach was able to find results closest to the reference non-inferior solutions. Figure 4.3 also shows that the Forrester method was able to generate points closer to the reference non-inferior solutions compared to GMOEA. These comparisons between all three approaches are best shown by comparing the best-case results for all three approaches. The qualitative verification of these visual results is shown in the quality metrics in Table 4.2. As established in Chapters 1 and 3: a lower value for generational distance indicates better performance, a higher value for hypervolume indicates better performance, a higher value for overall spread indicates better performance, and a lower spread metric value indicates better performance.

The proposed approach can perform the best for GD (Equation 1.13) (by having the smallest

Table 4.2: USV Case Study Quality Metric Results
(lower GD is better, higher HV is better, higher OS is better, lower spread is better)

Algorithm	Generational Distance ($\times 10^{-3}$)	Hypervolume ($\times 10^{-4}$)	Overall Spread	Spread
GMOEA	2.24 (0.503)	0.318 (0.436)	0.0202 (0.0244)	0.981 (0.0321)
Forrester	1.93 (0.472)	0.887 (0.772)	0.160 (0.0894)	**0.779 (0.126)**
Proposed Approach	**1.15 (0.395)**	**2.57 (0.526)**	**0.269 (0.111)**	1.09 (0.142)

value), HV (Equation 1.14) (by having the largest value), and OS (Equation 1.15) (by having the largest value) but gets outperformed by GMOEA and Forrester method for the spread metric (Equation 1.16). The reason why the proposed approach performed worse than GMOEA and Forrester method in terms of Deb's spread metric (Equation 1.16) is due to the proposed approach generating clusters of points. This is shown in the best case of the proposed approach where the expected system reliability is 0.735 and 0.736 in figure 4.3. The existence of these clusters shows that the proposed approach did not generate points uniformly along the non-inferior set of solutions that the approach was able to evaluate. Since GMOEA and Forrester methods did not have dense clusters of points, they were able to outperform the proposed approach in terms of deb's spread metric (Equation 1.16).

None of the tested approaches (GMOEA, Forrester method, and proposed approach) were able to obtain points that have an expected system reliability above 0.73675. This is the region where the reference non-inferior front transitions to a much steeper relationship between the expected system reliability and the expected speed. There is a linear trade-off between the expected system reliability and the expected USV speed in the reference non-inferior values with system reliability under 0.73675. This point shows a change in the system, where the system becomes much more sensitive to the expected speed of the USV after this point.

None of the three approaches (GMOEA, Forrester, and proposed approach) were able

to find points that were within the reference non-inferior solutions for the case study problem (Equation 4.8) after 250 function calls. This suggests that to identify better-performing solutions that approach the reference non-inferior solutions, a greater number of function calls would need to be permitted.

4.6 Summary

This chapter demonstrated the practical application of the proposed online surrogate-based multi-objective design optimizer on a realistic, computationally expensive engineering problem involving the control of a USV. Through this application, the objectives of maximizing the expected speed and expected system reliability of the USV were considered. Compared to GMOEA and the Forrester method, the proposed approach was able to outperform both in terms of closeness (with GD (Equation 1.13) and HV (Equation 1.14)) and diversity in regards to OS. However, the proposed approach showed its limitations in regards to its performance on the Deb's spread metric. This limitation was shown by the proposed approach being unable to evenly distribute solutions along the non-inferior front for the case study problem (Equation 4.8).

The results provided in this chapter demonstrate the proposed approach's ability to handle multi-objective computationally expensive complex engineering optimization problems. While the proposed approach has shown promising results with a limited number of function calls, further improvements could be made. These improvements could include allowing for more function evaluations to improve the quality of the results shown, as well as changing the structure of the proposed approach to include an adaptive exploration strategy to mitigate the clustering effect that was shown.

The next and final chapter will discuss the concluding remarks for this thesis. It will provide a summarization of the key findings, discuss the limitations of the work done so far, and outline possible directions for future work in the area of online surrogate-based multi-objective design optimization.

Chapter 5: Conclusion

This chapter presents the concluding remarks of this thesis, highlighting its primary contributions to the literature along with the proposed approach. It summarizes the findings from both the numerical test problems and engineering test problems while acknowledging the approach's limitations. Additionally, it explores potential directions for future research.

5.1 Summary

The motivation for this work stemmed from the computational expense associated with performing multi-objective design optimization for complex engineering problems. These engineering optimization problems are known to require time-consuming simulations as well as computational power in terms of hardware specifications. Based on an extensive literature review, it was discovered that there is an apparent gap in the limited knowledge regarding online surrogate-based multi-objective design optimizers that use generative models with implicit constraint knowledge influencing the training of the generative model. Most of the currently existing optimizers use surrogate models in place of each of the objective and constraint functions in the problem and then perform an optimization process/iteration using these surrogates. The other method that was common in the literature was to use a surrogate to generate possible sets of decision variables that are expected to perform well based on their fitness evaluation from the optimizer's

fitness function. Neither of these approaches embeds the knowledge gained from the individual constraints in the optimization problem to influence the GAN's generator to generate predicted feasible offspring samples as part of the GAN's generator training process.

From the limited literature, the gap that was shown served as the inspiration for the approach presented in this thesis. This proposed approach, presented in this thesis, works by embedding the feasibility prediction from the SVM constraint surrogate models with the discriminator model to influence the training of the GAN's generator model. By influencing the training of the GAN's generator model, the generated offspring from the GAN's generator would have the properties learned through the training process. Thus, the offspring generation would be influenced by the discriminator's ability to predict if a given set of decision variables is within a desirable region of the design space as well as having a higher likelihood of being feasible due to the influence from the SVM constraint surrogate models. While it was seen in the literature that there has been a similar approach, this approach (from the literature) used the discriminator to predict the feasibility of the given set of decision variables. The proposed approach uses an SVM surrogate model to predict the feasibility of a given set of decision variables. This is done because the SVM model would find a boundary line that maximizes the distance between the infeasible and feasible regions. Maximizing this boundary distance between the two regions balances the area/volume between the infeasible and feasible regions. This, in turn, provides a more consistent method for finding the actual feasible/infeasible boundary for the constraint.

The result of this analysis of the literature and theorized improvement created a proposed online surrogate-based multi-objective design optimizer that leverages GANs and SVM constraint models. The structure for this proposed approach is shown in Chapter 2. This chapter demonstrates the methodology for integrating the GAN and SVMs to learn different properties of the optimization

problem. A step-by-step procedure for how the optimizer functions for each iteration of the optimization process was discussed. Once the full optimization process for the proposed approach was discussed, a time complexity analysis was performed. A time complexity analysis was also performed on two other online surrogate-based multi-objective design optimizers, which are GMOEA and the Forrester method. From the time complexity analysis, it was shown that the proposed approach was able to have a computational advantage over the Forrester method. However, GMOEA had a slight computational advantage over the proposed approach. The slight computational advantage that GMOEA has over the proposed approach is caused by the presence of the SVM constraint surrogate models, which are not present in GMOEA.

Once the structure for the proposed approach has been established, Chapter 3 demonstrates how the proposed approach performs on a set of various optimization test problems. The performance of the proposed approach is compared to GMOEA and the Forrester method for these optimization test problems. A visual and quantitative comparison of these approaches was used to show the effectiveness of each of the different approaches. It was shown that the proposed approach was able to outperform GMOEA and the Forrester method consistently in terms of GD and HV. This proved that the proposed approach has a statistically significant advantage over GMOEA and the Forrester method in terms of closeness. In terms of diversity, the proposed approach was able to outperform GMOEA and the Forrester method for many of the optimization test problems. The diversity was evaluated using OS and Deb's spread metric. While the proposed approach was able to outperform GMOEA and the Forrester method, the proposed approach did not have the same level of advantage that was demonstrated with the closeness metrics. This difference in performance between closeness and diversity was likely caused by the structure of the traditional GAN used in the approach. It was proved that the proposed approach has a

scalable and competitive performance based on the various numerical optimization test problems used. This validates how effective the implementation of the SVM constraint surrogate models with a GAN-based online surrogate multi-objective design optimizer is.

To extend the evaluation, the proposed approach was tested on a realistic engineering multi-objective case study for determining the throttle control profile for a USV in Chapter 4. The objectives in this case study were to determine a throttle profile that would maximize the expected speed and expected system reliability for the USV during its mission. The performance of the proposed approach was compared to the obtained non-inferior set of solutions generated by GMOEA and the Forrester method. The approaches were compared using the same closeness and diversity metrics used in Chapter 3. It was shown that the proposed approach was able to outperform both GMOEA and the Forrester method in terms of closeness (shown through the GD and HV metrics). The proposed approach was able to outperform GMOEA and the Forrester method for one of the diversity metrics (which was OS). The proposed approach was not able to perform the best in terms of Deb's spread metric. This shortcoming was able to demonstrate the limitations of the proposed approach, since the proposed approach was not able to uniformly distribute the points along the obtained non-inferior set of solutions for the case study optimization problem.

5.2 Contributions

The primary contribution of this study has been the development of an online surrogate-based multi-objective design optimization algorithm structure. The proposed approach uses GANs and constraint-boundary-informed SVMs to solve computationally expensive black-box

constrained multi-objective optimization problems. A trained SVM constraint surrogate model is used to influence the GAN to generate offspring that are predicted to be feasible. This is done while maintaining the ability to generate possible solutions that are expected to perform well, based on the fitness evaluation of the previously sampled sets of decision variables.

The performance of the proposed approach was compared against two other algorithms from the literature (GMOEA [17] and Forrester method [11]). This comparison was carried out using four quality metrics on a set of numerical optimization test problems from the literature. Many papers in the current literature compare their approaches based on the accuracy of the results. They do this using metrics such as GD and HV. This overlooks the diversity aspect for evaluating the non-inferior solution set generated by the online surrogate-based multi-objective design optimizer.

Finally, all the algorithms were applied to a wide variety of numerical multi-objective optimization test problems from the literature, as well as a realistic case study focused on the optimization of the throttle control profile of a USV. The scalable performance of the proposed approach was shown through the numerical multi-objective optimization test problems, through the variations in the complexity of the test problems, the number of decision variables present, the number of objectives in the problem, and the number of constraints in the problem. This was also able to show how the proposed approach can be applied to realistic, complex engineering optimization problems. All the tested online surrogate-based multi-objective design optimizers were constrained to have the same number of function calls (the number of times the entire optimization problem was evaluated). This was done so that the quality metric values obtained from evaluating the approaches are comparable. Along with these tests, a time complexity analysis was performed on all the tested algorithms that were used as a comparison against

the proposed approach. This demonstrated how the proposed approach effectively scaled in computational complexity as the number of allowable function calls increased.

5.3 Limitations

The limitations of this work include the use of a traditional GAN as the surrogate model for generating offspring sample points. This is a limitation because traditional GAN structures are susceptible to mode collapse. The description of mode collapse is as follows: mode collapse occurs when the GAN's generator produces a limited variety of output results. These results lack diversity in the values contained. This was shown in the test problems in the results in Chapter 3, through the proposed approach scoring poorly for Deb's spread metric. A possible method for fixing this issue moving forward would be to implement a more advanced GAN model, such as WGAN-GP, which has a more complex modified training structure to mitigate problems of mode collapse.

A limitation of the current GAN implementation is the absence of transfer learning [53], which could have leveraged prior knowledge from related problems to accelerate the optimization process. However, this decision was intentional to maintain flexibility in the proposed approach. If a transfer learning strategy were implemented, this would require the assumption that the new optimization problem shares characteristics with previously encountered ones. By designing the GAN to learn entirely from scratch for each specific optimization problem, without relying on external priors, the method avoids introducing biases and removes the need to assume similarity with past problems.

Another limitation is the current structure of the proposed approach, where each constraint

requires an independent SVM constraint surrogate model. As the number of constraints in the problem grows, the cost of training the constraint SVMs grows with it. This could be mitigated by creating an adaptive constraint policy to train only the SVM constraint surrogate models that are likely to be active for the points along the Pareto front. This would save the computational expense since the constraints that are not active are likely to have a significant amount of slack, so the accuracy of the surrogate model for these constraints does not need to be as high as the active ones. By incorporating this type of system into the approach, the number of SVM constraint surrogate models that need to be trained significantly reduces.

5.4 Future Work

Future work that could be done based on the information presented in this thesis is the implementation of a more advanced GAN model, such as WGAN-GP, to be used to solve the issues encountered with mode collapse as discussed in Section 5.3. In addition to this, also discussed in Section 5.3, an adaptive constraint training selection scheme could be implemented to reduce the computational expense created by the training and updating of each SVM constraint surrogate model for each iteration of the optimizer.

Other changes that could be investigated in future work would be to change the function of the different components of the GAN and SVM constraint surrogate models. This could be done by using the discriminator to compare the predicted ranks of different points to find a set of points with a high-quality fitness value based on work shown in this thesis and [31]. The discriminator would be trained to determine if one set of decision variables can dominate another set of decision variables. By making this comparison, both the generator and discriminator components of the

GAN learn to generate and identify the characteristics of points that allow one set of decision variables to dominate another set of decision variables. This could be combined in the form in which the discriminator focuses on whether one set of decision variables can dominate another set of decision variables, looking at only the objective function performance. Then the SVM constraint surrogate models would be able to provide information on whether a given set of decision variables is likely to violate at least one constraint and be infeasible. This would reduce the required complexity of the discriminator component of the GAN. Both the discriminator's ability to predict if one set of points dominates another set and the SVM constraint surrogate model's ability to predict if at least one constraint violation has occurred would be combined in the GAN's generator training process.

Another possible route for future work is to explore how this approach could be applied to solve an inverse formulation for the optimization problem. The goal of this would be to analyze how changes in the objective and constraint function values affect the decision variables' values. This would provide an understanding of how sensitive the trade-offs are in the optimization problem, as well as reveal the extent to which specific constraints restrict the design space. For this to be implemented, techniques from explainable artificial intelligence [54] can be employed; however, further research is required to understand the complete details.

Since this thesis focused on applying the proposed approach to the engineering multi-objective optimization problem for determining the throttle control profile for a USV, other engineering multi-objective optimization case study problems could be investigated. This would provide information on how this type of approach performs in the context of a wider variety of multi-disciplinary optimization problems.

5.5 Potential Impact

The potential impact of this work is the ability to solve computationally expensive multi-objective optimization problems faster. This speed increase is based on the assumption that the computational cost of a single evaluation of the optimization problem outweighs the additional computational cost created by the optimization process. The proposed approach would provide more feasible solutions with a high likelihood of success. With this ability to solve computationally expensive multi-objective optimization problems with fewer function calls, these problems have a more practical implementation for engineering work.

Since the proposed approach is designed for a general constrained multi-objective optimization problem, there is a wide range of applications where this approach can be used. This would include (as stated in Chapter 1), optimizing the design of CFD models [3], optimizing the design of FEA models [4], and many more. In addition to this, the high computational cost directly correlates with the required hardware for solving the engineering optimization problem. By reducing the computational cost of solving the engineering optimization problem, the required specification of the hardware can be relaxed. This, in turn, would mean that cheaper hardware could be used to solve the same problem, reducing the financial costs associated with purchasing and operating the necessary hardware. This would also mean that in terms of potential users, this approach would be valuable to anyone interested in finding sets of design parameters for a product that should meet specific requirements while remaining competitive in the market, particularly in cases where the evaluation of the design parameters is computationally expensive.

Bibliography

[1] Arko Chatterjee, Shapour Azarm, and Katrina Groth. Online Surrogate Multi-Objective Design Optimization using Generative Adversarial Networks with Constraint Assistance. *Journal of Mechanical Design*, 2025. In Preparation.

[2] Indranil Hazra, Arko Chatterjee, Joseph Southgate, Matthew J. Weiner, Katrina M. Groth, and Shapour Azarm. A reliability-based optimization framework for planning operational profiles for unmanned systems. *Journal of Mechanical Design*, 146(5):051704, 05 2024.

[3] Jincheng Zhang and Xiaowei Zhao. Wind farm wake modeling based on deep convolutional conditional generative adversarial network. *Energy*, 238:121747, 01 2022.

[4] Jakub Kudela and Radomil Matousek. Recent advances and applications of surrogate models for finite element method computations: a review. *Soft Comput*, 26(24):13709, 12 2022.

[5] Laurens Bliek, Arthur Guijt, Rickard Karlsson, Sicco Verwer, and Mathijs de Weerdt. Benchmarking surrogate-based optimisation algorithms on expensive black-box functions. *Applied Soft Computing*, 147:110744, 11 2023.

[6] Huixiang Zhen, Wenyin Gong, and Ling Wang. Offline data-driven evolutionary optimization based on model selection. *Swarm and Evolutionary Computation*, 71:101080, 06 2022.

[7] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13(4):455–492, 12 1998.

[8] Sunghyun Cho, Youngjin Kim, Minsu Kim, Hyungtae Cho, Il Moon, and Junghwan Kim. Multi-objective optimization of an explosive waste incineration process considering nitrogen oxides emission and process cost by using artificial neural network surrogate models. *Process Safety and Environmental Protection*, 162:813–824, 06 2022.

[9] Huachao Dong, Jinglu Li, Peng Wang, Baowei Song, and Xinkai Yu. Surrogate-guided multi-objective optimization (SGMOO) using an efficient online sampling strategy. *KNOWLEDGE-BASED SYSTEMS*, 220:106919, 05 2021.

[10] Guodong Chen, Kai Zhang, Xiaoming Xue, Liming Zhang, Chuanjin Yao, Jian Wang, and Jun Yao. A radial basis function surrogate model assisted evolutionary algorithm for high-dimensional expensive optimization problems. *Applied Soft Computing*, 116:108353, 02 2022.

[11] Alexander Forrester, András Sóbester, and Andy Keane. *Engineering design via surrogate modelling: a practical guide*. Wiley, Hoboken, N.J, Chichester, 2008.

[12] John Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61, 03 1999.

[13] Xiaosong Du, Ping He, and Joaquim R.R.A. Martins. Rapid airfoil design optimization via neural networks-based parameterization and surrogate modeling. *Aerospace Science and Technology*, 113:106701, 06 2021.

[14] Dachuan Liu, Peng Hao, Tengfei Xu, Yingjie Zhu, Xuanxiu Liu, Bo Wang, and Gang Li. Intelligent optimization of stiffener unit cell via variational autoencoder-based feature extraction. *Struct Multidisc Optim*, 66(1):8, 01 2023.

[15] Wei Chen and Faez Ahmed. MO-PaDGAN: Reparameterizing engineering designs for augmented multi-objective optimization. *Applied Soft Computing*, 113:107909, 12 2021.

[16] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53, 01 2018.

[17] Cheng He, Shihua Huang, Ran Cheng, Kay Chen Tan, and Yaochu Jin. Evolutionary multiobjective optimization driven by generative adversarial networks (GANs). *IEEE Trans. Cybern.*, 51(6):3129, 06 2021.

[18] Yu Zhang, Wang Hu, Wen Yao, Lixian Lian, and Gary G. Yen. Offline data-driven multiobjective optimization evolutionary algorithm based on generative adversarial network. *IEEE Trans. Evol. Computat.*, 28(2):293, 04 2024.

[19] Da Teng, Yun-Wen Feng, Cheng Lu, Behrooz Keshtegar, and Xiao-Feng Xue. Generative adversarial surrogate modeling framework for aerospace engineering structural system reliability design. *Aerospace Science and Technology*, 144:108781, 04 2024.

[20] Zhenzhong Wang, Haokai Hong, Kai Ye, Guang-En Zhang, Min Jiang, and Kay Chen Tan. Manifold interpolation for large-scale multiobjective optimization via generative adversarial networks. *IEEE Trans. Neural Netw. Learning Syst.*, 34(8):4631, 08 2023.

[21] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *Statistics and Machine Learning*, page 1, 12 2017.

[22] Zhengyang Zhang, Han Fang, Zhao Xu, Jiajie Lv, Yao Shen, and Yanming Wang. Multi-objective generative design of three-dimensional material structures. *APL Machine Learning*, 1(4):046120, 12 2023.

[23] Honglei Cheng, Gai-Ge Wang, Liyan Chen, and Rui Wang. A dual-population multi-objective evolutionary algorithm driven by generative adversarial networks for benchmarking and protein-peptide docking. *Computers in Biology and Medicine*, 168:107727, 04 2024.

[24] Zhendong Guo, Haitao Liu, Yew-Soon Ong, Xinghua Qu, Yuzhe Zhang, and Jianmin Zheng. Generative multiform bayesian optimization. *IEEE Transactions on Cybernetics*, 53(7):4347, 07 2023.

[25] Chao Wang, Jing Zhang, Zhabang Zheng, and Zhushou Liang. Using generative adversarial networks for efficient constrained multi-objective optimization. In *Proceedings of the 2024 3rd International Conference on Frontiers of Artificial Intelligence and Machine Learning*, pages 135–138. ACM.

[26] Vikramaditya Jakkula. Tutorial on support vector machine (svm). *School of EECS, Washington State University*, 37(2.5):3, 2006.

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, page 8024. Curran Associates, Inc., 2019.

[28] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. Artwork Size: 21 p. Medium: application/pdf.

[29] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*. Wiley-Interscience series in systems and optimization. John Wiley & Sons, Chichester ; New York, 1st ed edition, 2001.

[30] J. Blank and K. Deb. pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497, 4 2020.

[31] Guodong Chen, Jiu Jimmy Jiao, Xiaoming Xue, and Zhongzheng Wang. Rank-based learning and local model based evolutionary algorithm for high-dimensional expensive multi-objective problems. Version Number: 4.

[32] Jianqing Lin, Cheng He, and Ran Cheng. Adaptive dropout for high-dimensional expensive multiobjective optimization. *Complex & Intelligent Systems*, 8(1):271, 02 2022.

[33] Sarjinder Singh. *Simple Random Sampling*, pages 71–136. Springer Netherlands.

[34] M. Tanaka, H. Watanabe, Y. Furukawa, and T. Tanino. GA-based decision support system for multicriteria optimization. In *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, volume 2, pages 1556–1561. IEEE.

[35] Shibin Qiu and Terran Lane. *Parallel Computation of RBF Kernels for Support Vector Classifiers*, pages 334–345.

[36] Mikhail Belyaev, Evgeny Burnaev, and Yermek Kapushev. Exact Inference for Gaussian Process Regression in case of Big Data with the Cartesian Product Structure, 2014. Version Number: 2.

[37] A. Osyczka and S. Kundu. A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. *Structural Optimization*, 10(2):94, 10 1995.

[38] Evolutionary multiobjective optimization: theoretical advances and applications.

[39] David A. Van Veldhuizen. Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations. ISBN: 978-0-599-28316-9 Publication Title: ProQuest Dissertations and Theses.

[40] C.M. Fonseca, L. Paquete, and M. Lopez-Ibanez. An improved dimension-sweep algorithm for the hypervolume indicator. In *2006 IEEE International Conference on Evolutionary Computation*, pages 1157–1163.

[41] Jin Wu and Shapour Azarm. Metrics for quality assessment of a multiobjective design optimization solution set. *Journal of Mechanical Design*, 123(1):18, 03 2001.

[42] John H. McDonald. *Handbook of Biological Statistics*. Sparky House Publishing, 3 edition.

[43] Seatemperature Info. World sea water temperatures: https://seatemperature.info/, 2023.

[44] Weather Spark. Climate reports with the weather by month, day, even hour: https://weatherspark.com/, 2023.

[45] The MathWorks Inc. MATLAB version: 9.13.0 (R2022b), Natick, Massachusetts, United States. https://www.mathworks.com, 2022.

[46] Ahmed Ouadha and Youcef El-Gotni. Integration of an ammonia-water absorption refrigeration system with a marine diesel engine: A thermodynamic study. *Procedia Computer Science*, 19:754, 2013.

[47] Time and Date AS. Climate & weather averages in San Diego, California, USA: https://www.timeanddate.com/weather/usa/san-diego, 2023.

[48] Scripps Institution of Oceanography. Voyager: How long until ocean temperature goes up a few more degrees?: https://scripps.ucsd.edu/news, 2023.

[49] J C Chesley. *Handbook of Reliability Prediction Procedures for Mechanical Equipment*. Naval Surface Warfare Center (NSWC) Carderock Division, Logistics Engineering Technology Branch, 1998.

[50] SINTEF. *Offshore Reliability Data Handbook: 6th Edition*. OREDA Participants, Det Norske Veritas (DNV), 2015.

[51] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825, 2011.

[52] Fernando Nogueira. Bayesian Optimization: Open source constrained global optimization tool for Python, 2014.

[53] Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *J Big Data*, 3(1):9, December 2016.

[54] Plamen P. Angelov, Eduardo A. Soares, Richard Jiang, Nicholas I. Arnold, and Peter M. Atkinson. Explainable artificial intelligence: an analytical review. *WIREs Data Min & Knowl*, 11(5):e1424, September 2021.