ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference
IDETC2016
August 21-24, 2016, Charlotte, North Carolina

IDETC2016-60125

MULTI-OBJECTIVE COMPOSITE PANEL OPTIMIZATION USING MACHINE LEARNING CLASSIFIERS AND GENETIC ALGORITHMS

Kayla Zeliff
Air Force Research Laboratory
Rome, NY, USA
kayla.zeliff@us.af.mil

Dr. Walter Bennette
Air Force Research Laboratory
Rome, NY, USA
walter.bennette.1@us.af.mil

Dr. Scott Ferguson[1]
North Carolina State University
Raleigh, NC, USA
scott_ferguson@ncsu.edu

ABSTRACT
Design spaces that consist of millions or billions of design combinations pose a challenge to current methods for identifying optimal solutions. Complex analyses can also lead to lengthy computation times that further challenge the effectiveness of an algorithm in terms of solution quality and run-time. This work explores combining the design space exploration approach of a Multi-Objective Genetic Algorithm with different instance-based, statistical, rule-based and ensemble classifiers to reduce the number of unnecessary function evaluations associated with poorly performing designs. Results indicate that introducing a classifier to identify child designs that are likely to push the Pareto frontier toward an optima reduce the number of function calculations by 75-85%, depending on the classifier implemented.

INTRODUCTION
As computational power increases, so too does the design space complexity associated with modern engineered systems. It is now common to have problem formulations where the number of possible design alternatives number in the billions or trillions. When identifying an optimal solution from a large design space for a computationally expensive problem, designers often have to make a choice between 1) incorporating time-saving, low-fidelity analytic estimations or 2) accepting the computational cost and/or risk missing a deadline by using a high-fidelity model. Problems involving large, and also discrete, design spaces include composite materials where the number of plies, orientation angle of each ply, and the material of each ply cause the design space to expand rapidly. Carbon nanotube growth is another example problem, as the catalyst, laser power, temperature and pillar location on the patch affect overall growth measurement.

Algorithmic advances in optimization have explored how optimal solutions can be identified in complex design spaces without enumerating and evaluating all (or a large number of) alternatives. Examples of these methods include Genetic Algorithms [1], Particle Swarm Optimization [2] and Simulated Annealing [3]. These heuristic methods do not require gradient information and use information from previously evaluated designs to guide the development of future solutions. Despite these advantages, however, heuristic algorithms still require a burdensome number of objective function evaluations to identify the optimal design.

Attempts to limit the number of required objective function evaluations have resulted in approaches such as Design Space Reduction [4], Surrogate-Based Optimization [5], and Classifier Guided Sampling (CGS) [6]. Limitations associated with each of these techniques include the exclusion of potential Pareto optimal designs due to the exclusion of certain design combinations, meta-model error, and a focus on single objectives. The complex optimization problems of today involve large design spaces, require high-fidelity modeling and rarely seek to identify a single objective, but instead require simultaneous optimization of several objectives.

This work tests the effectiveness of using different classifiers with a Multi-Objective Genetic Algorithm to generate a set of non-dominated optimal solutions. To do this, an additional objective of minimizing the number of expensive function evaluations is included. More specifically, a multi-objective composite optimization problem is solved using the combination of a Multi-Objective Genetic Algorithm (MOGA) with k-Nearest Neighbor, Naïve Bayes, Random Forest and Decision Tree classifiers individually. Previous attempts to solve the composite panel optimization problem studied in this work required upwards of 5,000 expensive function evaluations [7]. This work seeks to drastically reduce this number while maintaining similar, if not better, solution quality.

The remainder of this paper is organized as follows: the Background section provides relevant information to the current state-of-the-art pertaining to multi-objective optimization and the classifiers used to limit function evaluations; the Methodology section introduces the algorithm used to incorporate a classifier into the multi-objective optimization; the Problem Formulation section describes the composite material problem used to demonstrated the developed methodology; the Results section explores and discusses the take-aways gleaned from the developed

[1]Corresponding author

algorithm; and finally, the Conclusion section summarizes the effort and discusses avenues for future work.

BACKGROUND

Maximizing our efficiency at solving large scale complex problems requires the combination of optimization algorithm advancements and the integration of machine learning classification techniques. The following subsections discuss existing optimization and classification algorithms, applications and limitations related to function evaluation reduction for complex problems.

Optimization

Greedy and heuristic optimization algorithms such as Particle Swarm Optimization, Simulated Annealing and Multi-Objective Genetic Algorithms have proven effective at identifying optimal solutions for complex design spaces [8, 9]. Particle Swarm Optimization [2] mimics bird flocking, fish schooling and swarming theory where each particle or design string keeps track of its location in hyperspace relative to the overall fitness achieved so far [10]. Simulated Annealing is analogous with the physical process of annealing solids and constitutes a series of neighborhood searches to identify the optimal cost function [3]. A Multi-Objective Genetic Algorithm, the optimization algorithm used in this work, behaves in a similar manner to Darwin's theory of Natural Selection where stronger designs remain and breed while weaker designs do not [1].

As these optimization algorithms are based on evolutionary or heuristic principles, the process of arriving at a final solution is often slower than gradient-based approaches and can require a larger number of objective function evaluations. For optimization problems where the design space is small and fitness assessments are computationally inexpensive, the large number of evaluations is generally not a limiting factor. However, as the design space and the computational cost of fitness assessment grow in complexity, the large number of evaluations often required by greedy and heuristic algorithms limits their effectiveness of identifying the optimal set of designs. Deadlines may result in early termination of the algorithm, leaving designers with a sub-optimal set of designs for exploration.

Design Space Reduction, Surrogate-Based Optimization and Classifier Guided Sampling have been introduced as approaches capable of identifying optimal designs in a reduced number of objective function evaluations for complex design spaces. Design Space Reduction limits the upper and lower bounds on design variables to reduce the number of feasible designs for potential evaluation [4]. Surrogate-Based Optimization involves defining, estimating, and validating a meta-model, replacing expensive function evaluations with a surrogate mathematical function [5, 11, 12]. Classifier Guided Sampling uses a classifier, trained by evaluated designs to identify optimal designs with fewer expensive function evaluations until predefined termination criteria are met [6]. For a detailed review of methodologies to handle computationally expensive optimization problems, see [12].

Approaches to limit the number of objective function evaluations using machine learning techniques have been applied to several optimization problems. For example, in [13], a Bayesian Network Classifier was applied to a series of interconnected sub-problems resulting from the decomposition of a UAV wing design problem. The results indicated that the classifier successfully identified the satisfactory regions of the design space given performance constraints. Similarly, in [14], a support vector machine was used to define a valid input to the predictive model. CGS was incorporated into a single objective Genetic Algorithm and applied to an autonomous microgrid design problem to determine appropriate layouts for power maintenance during adverse events [15]. Based on a MOGA, the P3GA methodology presented in [16] uses the support vector domain description (SVDD) to determine the predicted feasible set for the identification of the parameterized Pareto frontier, which is used for modeling technical capabilities.

While successfully reducing the need for expensive function evaluations, Design Space Reduction, Adaptive Sampling and Classifier Guided Sampling have limitations. Design Space Reduction may result in only portions of the true Pareto set being identified because certain combinations of design variables are removed from the feasible design space [4]. Adaptive Sampling relies on the accuracy of meta-models throughout the duration of the optimization [5]. The accuracy of the meta-model, if one is at all attainable, determines the selection of new designs and may result in the removal of an acceptable design or continue to keep a subpar design. Applications of Classifier Guided Sampling do not explore the applicability of different classifiers to problem sets and instead use the same machine learning classifier for each optimization problem [17].

To address these limitations, this work seeks to combine optimization through a MOGA and aspects of Adaptive Sampling and Classifier Guided Sampling. The following subsection describes the relevant background information regarding the use of classifiers as predictors for the identification of designs likely to improve the Pareto frontier.

Classification

In this work, machine learning classifiers serve as a proxy for expensive function evaluations and help identify promising areas of the design space to explore. Machine learning classifiers are aptly positioned for this application because they create abstractions of historical data and look for underlying patterns and trends to make future predictions [18]. Although the classifier will likely not be a perfect representation of the true system, the belief is that the classifier can learn enough about the system to identify promising areas that warrant a more intensive search.

There are several categories of classifiers to choose from, and the correct classifier to use is not immediately apparent and is likely problem dependent. In this work one classification technique is chosen from each of the following: instance-based classifiers, statistical classifiers, rule-based classifiers, and ensemble classifiers.

k-Nearest Neighbors (kNN)

Instance-based classifiers do not create a model of the training data to perform classification. Instead, instance-based classifiers rely on the training instances themselves to assign class labels to new instances. For example, with k-Nearest Neighbors (kNN) [19], an unlabeled instance is classified as the most prevalent class observed in the nearest k training instances. This is a very simplistic approach and works well when instances belonging to different classes are well separated. However, instance-based classifiers require a suitable distance metric to provide meaningful classifications.

To illustrate kNN, an example dataset with two dimensions (horizontal and vertical position) and two classes (black and red) can be constructed. If k = 1 for this dataset, as shown in Figure 1, a new point (X) would be classified as "black". In Figure 2 where $k = 3$, the new point would be classified as "red".

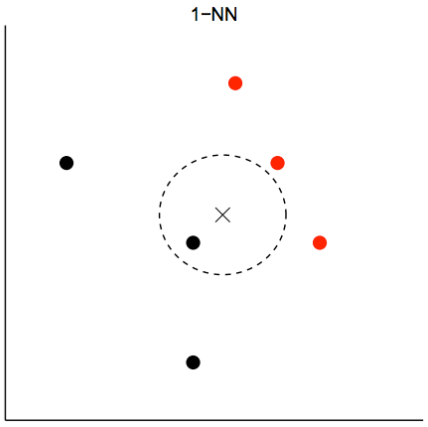

Figure 1. For $k = 1$, the new instance, represented by the "X", will be classified as "black"

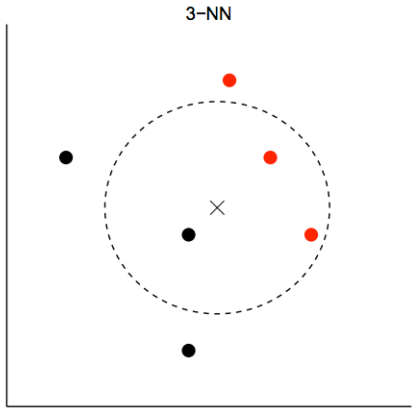

Figure 2. For $k = 3$, the new instance, represented by the "X", will be classified as "red"

Naïve Bayes

Naïve Bayes is a statistical classifier that requires simple prior probability calculations to classify an instance, and has been shown to have good accuracy for a wide variety of classification problems [20]. The probability prediction of naïve Bayes operates under the assumption that the effect of an attribute value on a given class is independent of the values of the other attributes [18]. This assumption allows the required calculations of the classifier to be simple, but performance is better when an instance's attribute values affect its class outcome with higher levels of independence.

In use, naïve Bayes can classify an instance with n attribues, $X = (x_1, x_2, \dots, x_n)$, as belonging to one of m possible classes. Instance X is predicted to belong to class C_i if and only if

$$P(X|C_i)P(C_i) > P(X|C_j)P(C_j) \; for \; 1 \le j \le m, j \neq i \tag{1}$$

where $P(X|C_i)$ and $P(C_i)$ are be estimated from the training data. Given the naïve assumption that the attributes are conditionally independent of one another, $P(X|C_i)$ can be calculated as

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) \tag{2}$$

Decision Trees

Rule-based classifiers classify new instances by following a series of "if-then" rules applied to the different attributes of an instance. Decision trees are a good example because their tree structure is easily deconstructed into rules. Decision tree algorithms induce their tree in a top-down manner by selecting attributes one at a time and splitting the training instances into groups according to the values of their attributes. The most important attribute is selected as the top split node, the next most important attribute is considered at the next level, and so forth. For example, in the popular C4.5 algorithm [21], attributes are chosen to maximize the information gain ratio in the split. This is an entropy measure designed to increase the average class purity of the resulting subsets of instances as a result of the sequential splits. Decision trees perform well when an instance's attributes actually have a hierarchical structure in regard to determining class label. However, they are prone to overfitting. That is, discovering structure that is not intrinsic to the true problem [18].

Figure 3 shows an example decision tree built from weather observations where the class records if tennis was played on that day. In this example tree, tennis is played if it is overcast or if it is sunny with low humidity.

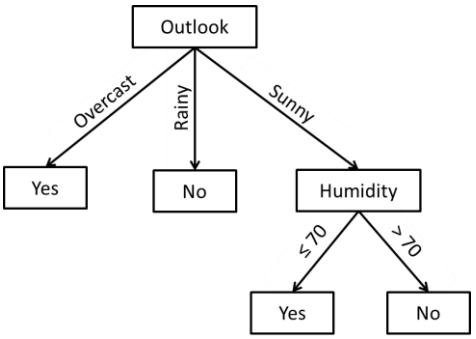

Figure 3. Example decision tree to determine appropriate weather for playing tennis

Random Forest

Ensemble classifiers use many classifiers to make a single prediction. For example, random forest is an ensemble technique that creates a large number of decision trees and uses a voting mechanism to assign a class to an unlabeled instance. In random forest, a single decision tree is created using a random subset of the training data's attributes and has additional randomness introduced through bagging, or sampling the original training data with replacement [22]. It has been found that random forest can achieve improved accuracy in comparison to decision trees.

To summarize, selection of the correct classifier is likely problem dependent. However, each classifier has its own strength. Instance based classifiers work well when the different classes are well separated in the design space. Naïve Bayes works well with high levels of independence between the problem's attributes. Decision Trees work well when different classes can be isolated by creating hierarchical splits of the data. Finally, Random Forests can improve upon the performance of single-model classifiers. The goal of this paper is to understand the effectiveness of each classifier when it is combined with a multi-objective genetic algorithm. Of particular interest is the ability to reduce the number of objective function evaluations, the quality of the non-dominated Pareto frontier, and the ability of the classifier to correctly predict design performance.

METHODOLOGY

The methodology presented in Figure 4 was created to assess the effectiveness of the different classifiers. An initial population is first created using a random initialization approach, as in NSGA-II [1]. These initial designs are then evaluated using calls to the objective function, and the non-domination ranking (number of points that dominate the point in question) of each point is calculated. If the termination criterion of the MOGA is not satisfied, the highest ranked designs are selected and new designs are created using crossover and mutation. Crossover is achieved using tournament selection and arithmetic crossover with a rate of 75%, while mutation occurs at a rate of 3% by increasing or decreasing the design variable. For termination criteria, either the number of generations evaluated was fixed, or the hypervolume of the Pareto frontier had to remain constant for 3 consecutive generations.

After new children are created each generation, the next step is to determine which designs should be evaluated and which designs should be discarded without evaluation. To aid in this process, the classifier is retrained using the information from all previously evaluated designs. The classifier is then used to predict which children are likely to achieve a non-domination ranking within a certain value of the current set of non-dominated points. Children predicted to be within this range are evaluated using calls to the objective function or identification of the function value from a repository of previously evaluated designs.

In this work, the accepted non-domination rank was reduced as a function of generation count. Initially, designs with a non-domination rank of 5 or less were marked as acceptable. In subsequent generations, as shown in Table 1, the acceptable non-domination rank was reduced until only designs predicted to be non-dominated were evaluated. The classifiers were then retrained by including the newly evaluated designs in the training set.

Allowing a greater number of designs to be evaluated in early generations was desired as it would give additional training information to the classifier. It was hypothesized that this strategy would allow the non-dominated frontier to expand more quickly. Reducing the acceptable non-domination rank incrementally assumes a sufficient number of objective function calls have occurred to supply the classifier with the necessary training set diversity.

Table 1. Accepted Rank Relative to Generation Count

Generation	Accepted Non-Domination Rank
1-2	5
3-5	4
6-8	3
9-11	2
12+	1

Classifier performance is compared in terms of average precision and recall [23]. In this context precision relates to the percentage of designs predicted by the classifier to be "good"" that are actually "good". Alternatively, recall encapsulates the percentage of actually "good" designs that are predicted to be "good". Both recall and precision are calculated by evaluating all designs generated by the MOGA, although only the designs predicted by the classifier to be "good" are used by the MOGA. For this work, a useful classifier should achieve a high level of both precision and recall.

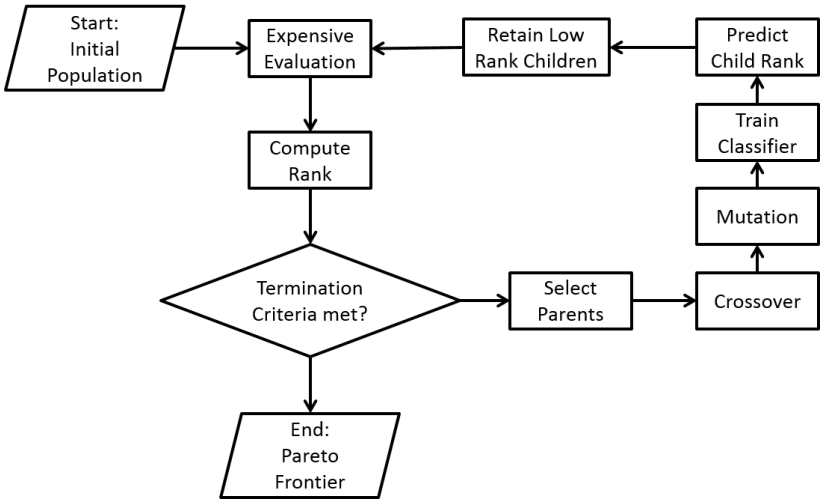

Figure 4. Methodology for the Combination of MOGA and Machine Learning Classifiers

PROBLEM FORMULATION

The test problem used to assess classifier performance is a pressurized fuselage skin modeled as a two-dimensional plate. Pressure applied to the panel induces loads in both the axial and hoop directions and a compressive load may exist that results in fuselage bending. The composite in this analysis is a square 1 inch by 1 inch graphite epoxy composite with a constant thickness of 0.01 inches and simply supported edges. The material properties are given in Table 2.

Table 2. Composite Material Properties

Material Properties	Values
E_1	30×10^6 psi
E_2	0.75×10^6 psi
ν_{12}	0.25
G_{12}	0.375×10^6 psi
X_t	150×10^3 psi
Y_t	6×10^3 psi
S	10×10^3 psi
X_c	100×10^3psi
Y_c	17×10^3 psi

A single load scenario is considered consisting of uniaxial tension and transverse compression. This load scenario is depicted in Figure 5. Buckling failure is examined for compressive loads.

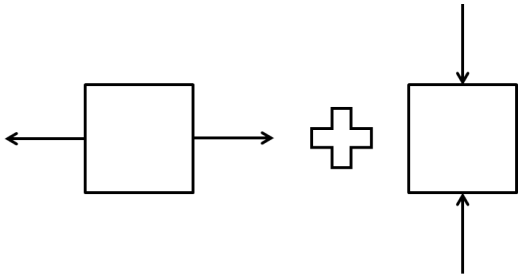

Figure 5. Uniaxial Tension and Transverse Compression

The objective functions associated with the composite panel consist of maximizing the uniaxial tension and transverse compression before buckling occurs. Each composite panel consists of 8 plies oriented at angles in intervals of 10 degrees between -90 degrees and 90 degrees, inclusive. The typical manufacturing constraints of symmetry and balance are not strictly enforced through problem formulation. Equation (3) depicts the composite material optimization problem in standard form.

$$\begin{aligned} &\text{Maximize:} && F_1(X), F_2(X) \\ &\text{Subject to:} && x_i \bmod 10 = 0 \\ & && 90 \le x_i \le -90 \end{aligned} \tag{3}$$

In Eq. (3), F_1 and F_2 are the competing objectives of uniaxial tension and transverse compression. They are functions of the design string X, where $X = \{x_1, x_2, \ldots, x_8\}$. The total number of design combinations possible exceeds 16 billion, making this a suitable problem for demonstrating the effectiveness of a classifier when used to augment MOGA performance.

The initial population was set to, and remained fixed at, 100 designs. Tournament selection was used to obtain 75% of the population for crossover. Arithmetic crossover was implemented with $\lambda = 0.4$ and the resultant design rounded to the nearest 10. Mutation involved randomly increasing or decreasing 3% of the child design plies by ±10 degrees.

A one-size-fits-all solution for determining MOGA convergence has yet to be discovered. Possible termination criteria include a set number of function evaluations, a set number of generations, or stagnation in the hypervolume of the non-dominated frontier for a defined number of generations. To accurately compare the Pareto frontiers produced by the control and the ones using the different classifiers, a set number of function evaluations could not be used as termination criteria as

it may not produce the true Pareto frontier. Similarly, using a set number of generations makes for an easy comparison of function counts, but runs the risk of not producing the true Pareto frontier. Instead, the hypervolume of the Pareto frontier was calculated each iteration and used as the convergence measure for the first study. A MOGA was considered to have converged if the hypervolume remained constant for 3 consecutive generations.

As discussed in the previous section, the kNN, naïve Bayes, C4.5 decision tree, and random forest classifiers were implemented. Specifically, kNN was implemented with k = 1 and a Euclidean distance metric. Each random forest consisted of 100 trees. The attributes of the classifiers' training data are the orientation of each composite layer, with the class label corresponding to the proximity of a design to the Pareto frontier (the non-domination rank). Each classifier is retrained every generation after evaluating the designs predicted to be "good" in accordance with the schedule presented in Table 1.

RESULTS

Pareto frontier results are compared in terms of function evaluations where each function evaluation consists of all objective function calculations for a specific design. That is, for the purposes of the composite material problem, a single function evaluation consists of the evaluation of the transverse buckling and uniaxial tensile loads of the design. In this way, the methodology presented in Figure 4 may be compared to optimization problems with larger numbers of objectives.

The methodology presented in Figure 4 was implemented with the previously mentioned classifiers of kNN, naïve Bayes, Decision Tree, and Random Forest individually and run one time to convergence. For comparison, a MOGA was run without a classifier to provide a function count and a Pareto frontier that was reached after 5,861 function evaluations.

Classifier performances are compared in terms of average precision and recall [23]. Recall that:

- Precision: percentage of designs predicted by the classifier to be "good"" that are actually "good"
- Recall: percentage of actually "good" designs that are predicted to be "good"

Table 3 provides the number of function evaluations, the average precision, and the average recall for each method. Both recall and precision are calculated by evaluating all of the MOGA suggested designs, although only the designs predicted by the classifier to be "good" are actually used in the procedure and added to the function count. For this work, a useful classifier should achieve a high level of both precision and recall. The following subsections discuss the results pertaining to each classifier on an individual basis.

Table 3. Function Evaluations, average and standard deviation of Precision and Recall of each Method

Classifier	Function Evals	Precision (Mean)	Precision (SD)	Recall (Mean)	Recall (SD)
None	5861	-	-	-	-
kNN	868	69%	10%	75%	10%
Naïve Bayes	1449	41%	20%	85%	13%
Decision Tree	676	67%	12%	55%	15%
Random Forest	918	82%	11%	79%	13%

k-Nearest Neighbor

The blue data points in Figure 6 depict the kNN Pareto frontier. Unlike the frontiers generated through Decision Trees, Naïve Bayes and Random Forest, the kNN frontier spans the entire length of the *no classifier* result. The convex behavior observed between the transverse compression values of 45 and 55 lbf are accurately represented through the identified kNN frontier. However, between 55 and 60 lbf, the kNN frontier demonstrates additional convex behavior not found in the *no classifier* frontier. The inherent evolutionary behavior of a genetic algorithm removes the ability of an operator to exactly reproduce results. Therefore, altering the termination criteria may result in the removal of certain convexities within the Pareto frontier, but may also produce additional behaviors not currently present.

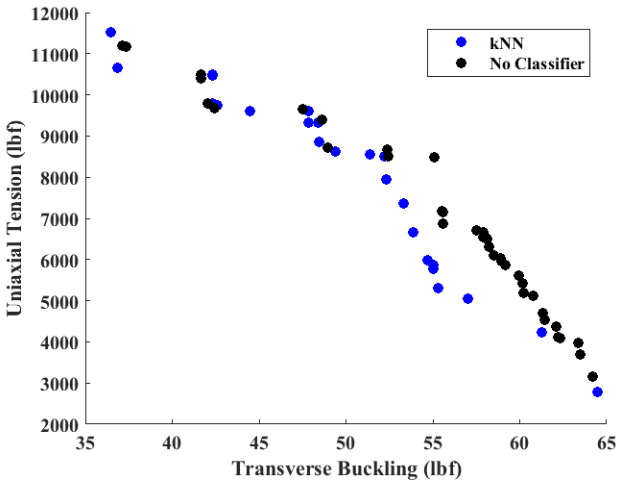

Figure 6. kNN and *No classifier* Pareto Frontiers

In terms of function evaluations, the blue kNN Pareto frontier was produced in 868 function evaluations. This is an 85% reduction in function evaluations when compared to the number of function evaluations required to produce the *no classifier* Pareto frontier. While the exact requirements of an optimal design are at the discretion of the designer, the similarities of the *no classifier* and kNN frontier indicate that the introduction of a classifier in the initial evaluation of child designs for

complex optimization problems results in comparable results for a significant reduction in function evaluations and by extension, time.

When considering the predictive ability of the kNN classifier in the CGS procedure, its average recall is 75% and average precision is 69%. This means that of the "good" designs suggested by the MOGA, 75% of them are evaluated in the procedure, and 69% of the evaluated designs are actually "good". Although this is not perfect, it does capture enough "good" designs to push the Pareto frontier, and obviously limits the number of wasted function evaluations. It is perhaps not surprising that kNN performs well for this problem as "good" designs may have similar orientations. However, performance of the kNN classifier may be improved by deriving a distance measurement that incorporates the angular differences between the layers.

Naïve Bayes

The green data points in Figure 7 depict the Pareto frontier generated through the incorporation of the naïve Bayes classifier into the selection of child designs for evaluation. With the exception of a few data points, the naïve Bayes frontier falls slightly behind the *no classifier* Pareto frontier used for comparison. The convex behaviors apparent in the *no classifier* frontier are not reflected in the Naïve Bayes frontier. The naïve Bayes frontier has difficulty identifying the middle of the frontier space where transverse compression is between 45 and 55 lbf. The *no classifier* frontier contains several convex regions that are not apparent in the naïve Bayes frontier. As with kNN, modifications to the termination criteria may result in the naïve Bayes frontier expanding encompass the convex nature of the *no classifier* frontier.

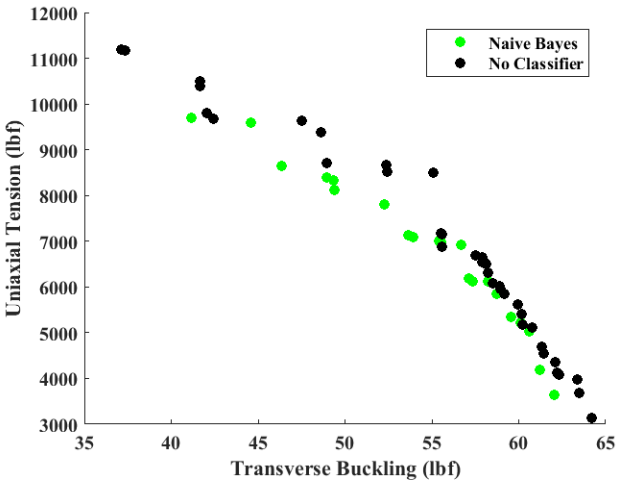

Figure 7. Naïve Bayes and *No classifier* Pareto Frontiers

The algorithm converged after 1449 function evaluations. While this is a larger number of function evaluations when compared to kNN, naïve Bayes still resulted in a 75% reduction in function evaluations from the *no classifier* option.

However, when analyzing the average precision and recall of the naïve Bayes classifier, it is not surprising that the number of function evaluations is high. This is because the average recall is 85%, while the average precision is only 41%. Therefore, many of the function evaluations are being wasted on "bad" designs that do not help define the Pareto frontier. Still, the fact that naïve Bayes is not a great classifier for this problem is understandable. Naïve Bayes assumes independence between the layers, which is certainly not the case as was identified in [7] by the balanced and symmetric designs composing the Pareto frontier despite a lack of enforcement through problem formulation.

Decision Tree

In Figure 8, the red data points identify the Pareto frontier produced through the incorporation of the Decision Tree classifier into the methodology presented in Figure 4. The Decision Tree instantiation captures an area of the Pareto frontier not identified by the *no classifier* option. This indicates that no frontiers have successfully converged upon the true Pareto frontier for the composite panel optimization problem. While impressive in its ability to explore an additional region of the design space, the Decision Tree implementation severely struggled to explore the region of the solution space where the uniaxial tensile load is high and the transverse compression is low. The left tail of the frontier falls nearly 3.00 lbf less than the solutions identified by kNN and the *no classifier* option and there is a relatively large space of approximately 12 lbf between the left tail of the Decision Tree frontier and the next point on its frontier when traveling along the *x*-axis.

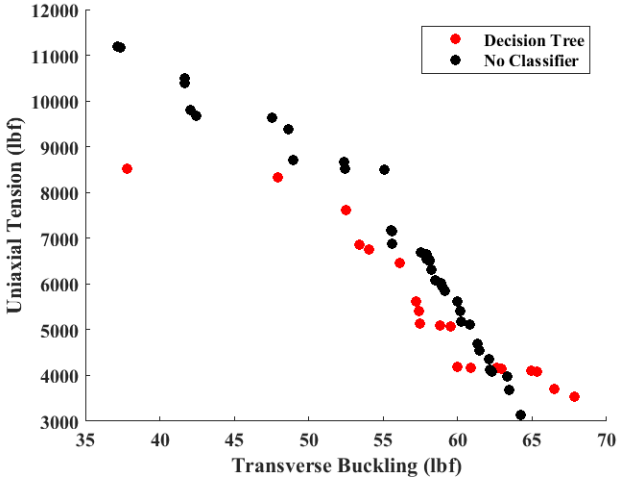

Figure 8. Decision Tree and *No classifier* Pareto Frontiers

The Pareto frontier identified through the incorporation of the Decision Tree classifier met the termination criteria in 676 function evaluations. This is a reduction in expensive function evaluations of slightly more than 88%. However, except in the case of relatively large transverse compression and small uniaxial tension, the Decision Tree classifier provided a poor result compared to the *no classifier* frontier. As with kNN and naïve Bayes, an alteration of the termination criteria may result

in the Decision Tree classifier option producing a frontier most similar to that of the *no classifier* option in much fewer function evaluations.

The poor performance of the MOGA with the decision tree classifier is likely due to its low average recall of only 55%. Meaning, almost half of the "good" designs suggested by the MOGA go untested in the procedure. It is possible that this low recall is due to the imbalance of the training data. For example, in the last iteration, the training data contained 59 "good" designs, and 607 "bad" designs (note that there are more "good" designs than points on the Pareto frontier because the MOGA may suggest designs that already exist in the training data, and if evaluated will be included again). In this scenario the decision tree may be overfitting to the data in favor of the "bad" class. It may be possible to increase the recall of the decision tree by increasing the value of the "good" class in a cost sensitive approach. However, this would likely decrease the tree's precision.

Random Forest

In Figure 9, the cyan data points represent the Pareto frontier produced through the use of a Random Forest classifier. With the exception of the extreme ends, the Random Forest frontier mimics the behaviors of the *no classifier* Pareto frontier. This includes the large number of data points present toward the larger transverse compression values and the convex behavior found between 45 and 55 lbf on the *x*-axis. While the behaviors of the Pareto frontiers are similar, the data points with larger transverse compression values reside much closer to the *no classifier* Pareto frontier than those with smaller transverse compression values. This behavior is not unique to the Random Forest frontier, but is also present in the naïve Bayes and Decision Tree frontiers.

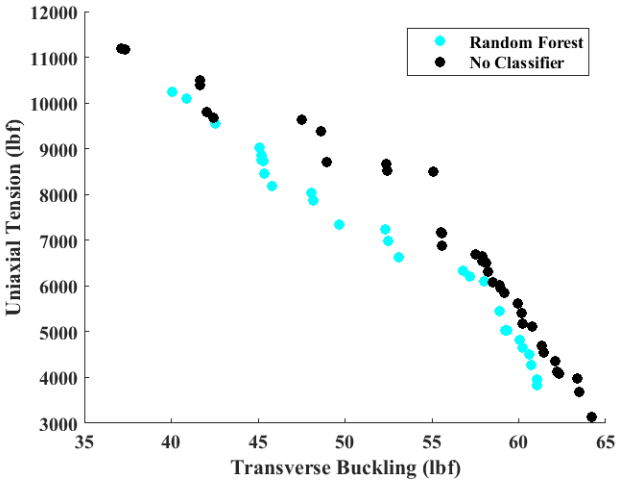

Figure 9. Random Forest and *No classifier* Pareto Frontiers

In terms of function evaluations, the cyan Random Forest frontier was produced in 918 function evaluations. This is a reduction of 84% from the *no classifier* MOGA. The small number of function evaluations and ability of the classifier to produce a frontier with similar behavior to that of the *no classifier* option indicates the Random Forest classifier may be well suited for complex composite optimization problems.

Evaluating the average precision and recall of the Random Forest classifier reveals that it performs best, with the highest values of both precision and very good recall. Although a Random Forest is composed of decision trees, the randomness and voting aspect of the algorithm help avoid the problem of overfitting [22]. One additional benefit of this classifier is that it may be possible to extract some understanding of the underlying problem by analyzing the structure of the classifier.

Combined Results

As depicted in Figure 10, the combination of the Pareto frontiers closely follows or exceeds the behavior of the *no classifier* frontier. The kNN frontier best covers the lower transverse compression values while the Decision Tree frontier encapsulates a region of the solution space not identified by the *no classifier* frontier. Further, the sum of the function evaluations required by all the classifiers to identify a Pareto frontier totaled less than that required by the *no classifier* option.

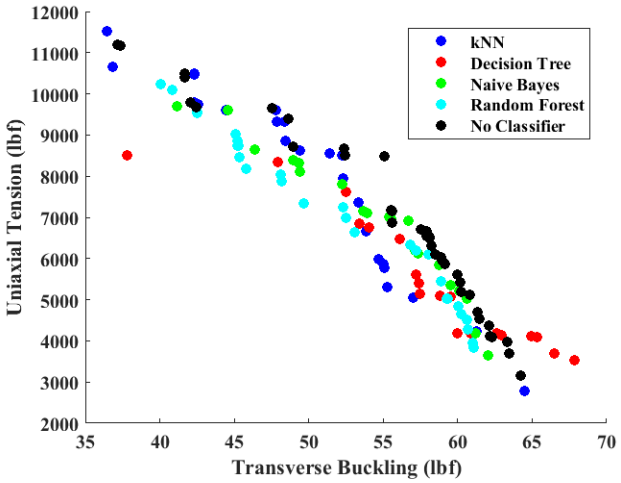

Figure 10. Composite Pareto Frontiers

With this information, Figure 11 depicts the identification of the non-dominated points of the combination of all four classifier frontiers plotted on the same axes as the *no classifier* Pareto frontier. For the smaller transverse compression loads, the Pareto frontiers have similar convex behavior. However, as the transverse compression load increases, the behavior of the frontiers begins to diverge with the *no classifier* frontier following the expected curvature of a Pareto frontier and the classifier frontier extending out into regions of the design space unidentified by the *no classifier* option.

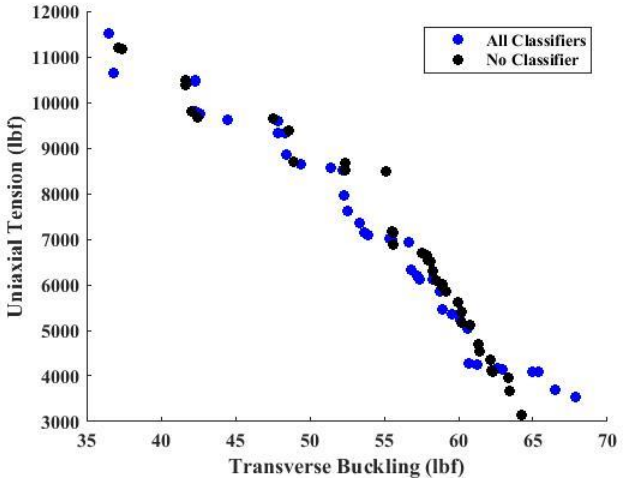

Figure 11. Combined Pareto Frontier

Since each frontier appears to contain a "better" solution in some regions of the solution space than the other, it is not a binary matter of stating one frontier is superior to another. From the perspective of hypervolume, the combined classifier frontier has a larger hypervolume as shown in Table 4. This indicates that the area of the solution space dominated by the classifier option is larger than that of the *no classifier* option.

Table 4. Hypervolume Comparison

No classifier	**All Classifiers**
1.95*E*9	1.98*E*9

While the hypervolume of the classifier frontier may be larger, it is clear from Figure 11 that the *no classifier* frontier contains several points that dominate the classifier frontier. Table 5 summarizes the results of comparing the Pareto frontiers by the number of designs dominating the designs of the other frontier. The *no classifier* frontier consists of 37 total designs, 10 of which are dominated by designs on the Pareto frontier generated with the combined classifier results. On the other hand, the classifier frontier consists of 40 designs and 26 of them are dominated by designs in the *no classifier* Pareto frontier.

Table 5. Dominated Points Comparison

	Total Number of Designs	**Dominated Points**	**Percentage**
No classifier	37	10	27%
Classifier	40	26	65%

These results indicate that while the classifier option may dominate a larger area of the solution space, the *no classifier* option dominates over half of the designs generated by classifier methodology. In further defense of the classifier generated frontier, the resultant frontier was generated in 3,911 function evaluations. This is nearly 2,000 less function evaluations than the *no classifier* Pareto frontier.

Function Count Convergence

When run only once using the hypervolume as termination criteria, the introduction of a classifier into the selection process of a MOGA greatly saves computational cost in terms of function evaluations. However, MOGAs are inherently stochastic methods and repeated runs will not necessarily produce the same results.

Rather than using hypervolume as a convergence metric, a function count convergence criteria was also used to compare the performance of the different classifiers. The algorithm was set to converge after 500 function evaluations, but was allowed to complete the generation it was evaluating when the functional count was reached. This number of function calls was selected to explore the performance of the classifiers during the "earlier" generations.

Figure 12 displays the resultant Pareto frontiers created from the aggregation of 5 runs of each classifier or no classifier option. As expected, the classifier options outperform the *no classifier* option when the number of function counts is limited. The decision tree classifier appears to outperform the other classifier options.

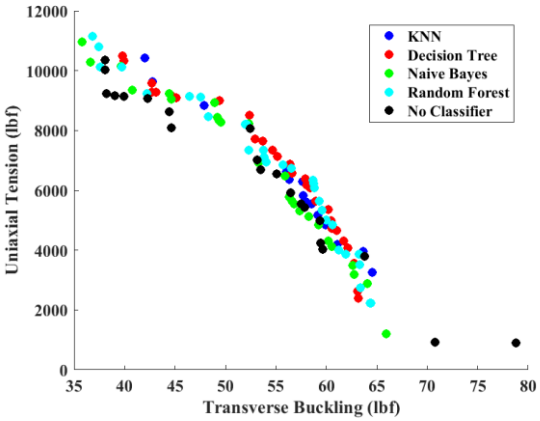

Figure 12: Function Count Convergence Pareto Frontiers

To better understand the results depicted in Figure 12, the average hypervolume and number of function evaluations are depicted in Table 6. Despite the visual appearance that the Decision Tree classifier outperforms the others, when considered in terms of hypervolume, the Random Forest classifier covers a greater portion of the solution space.

Table 6: Function Count Convergence Comparison

Classifier	**Function Count**	**HyperVolume**
kNN	529	1.37E9
Decision Tree	515	1.33E9
Naïve Bayes	564	1.21E9
Random Forest	523	1.40E9
No Classifier	531	1.30E9

Generation Convergence

To further evaluate the procedure discussed in this work, the convergence criteria was set to a hard limit of 15 generations and each MOGA was run 5 consecutive times and the aggregate Pareto frontier obtained.

Figure 13 depicts the aggregate Pareto frontiers of each classifier and the *no classifier* option after 15 generations. For the smaller transverse compression values, the frontiers appear to bunch together with the *no classifier* option slightly ahead. However, as the transverse compression values increase, there is slight separation between the frontiers with the *no classifier* option producing a data point not found in any previous runs.

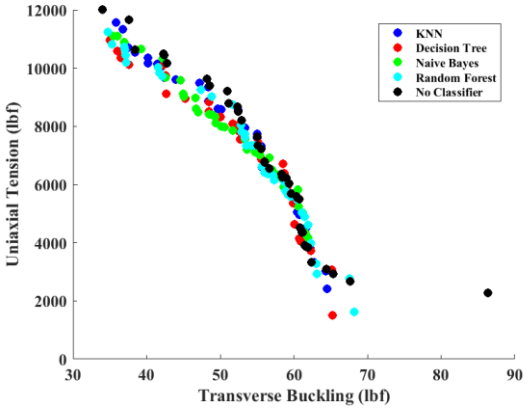

Figure 13. Generation Convergence Pareto Frontiers

While the Pareto frontiers shown in Figure 13 appear similar, the greatest difference is the number of function evaluations required to obtain the result. Table 7 displays the average function count across each of the 5 runs for each algorithm. From the table, even the worst performing Naïve Bayes algorithm produces the frontier shown in Figure 13 in 28% fewer function evaluations than the *no classifier* option. When considering the hypervolume of the frontiers produced, the *no classifier* option is largest with kNN, Decision Tree and Random Forest relatively close to each other. Naïve Bayes also performs the worst in terms of hypervolume calculation.

With altered convergence criteria and repeated runs of the procedure, Random Forest no longer performs the best. Instead, the Decision Tree classifier has the largest average function count and hypervolume. It is possible that further investigation into the precision and recall of the classifiers may indicate a greater number of training data is required for performance to improve.

Table 7. Generation Convergence Function Count Comparison

Classifier	Function Count	Hypervolume
kNN	811	5.1E8
Decision Tree	560	5.9E8
Naïve Bayes	1474	3.9E8
Random Forest	651	5.7E8
No Classifier	2035	2.0E9

CONCLUSION

Due to the sheer size of modern design problems, specialized optimization methods that minimize the number of required samples (or evaluations) need to be developed. Although methods such as Genetic Algorithms, Particle Swarm Optimization, and Simulated Annealing drive at this point, they often require too many samples for large and even mildly time sensitive projects. This work has shown that a machine learning classifier can be used to reduce expensive sampling by serving as a proxy for function evaluations in a Multi Objective Genetic Algorithm.

For the problem of composite materials, a MOGA with the Random Forest classifier was able to reduce function evaluations by as much as 84% in comparison to a MOGA with *no classifier*, while still maintaining an acceptable Pareto frontier. In the worst case scenario, one where the results from all classifier enhanced MOGAs are combined, the number of function evaluations is still appreciably reduced. Indeed, combining a machine learning classifier with a MOGA can improve the feasibility of performing optimization for complex design problems.

To further improve the interaction of machine learning classifiers and MOGAs it is necessary to customize classification models for specific applications. For example, the Naïve Bayes classifier, which assumes independence between attributes, was clearly ill-suited for the problem of composite materials. Other classifiers, such as kNN and Random Forest, performed well, but could still be improved.

One approach to improve classifier performance could be to create additional attributes that better describe the data. For the example of composite materials, it may make sense to include attributes that capture the orientation of layers with respect to their neighbors. Although humans can quickly calculate this difference, it is unlikely that a simple machine learning classifier would be able to capture and exploit such a relationship from basic attributes. Another approach to improve classifier performance could be to perform cost sensitive learning. Cost sensitive learning could put more weight on the important class and increase the classifier's recall rate, although this this typically comes at the cost of precision.

FUTURE WORK

While the purpose of this work was to demonstrate the large reduction in function evaluations through the incorporation of a classifier into the selection process of a MOGA, the results of the Decision Tree MOGA indicated that none of the Pareto frontiers produced reflected the true Pareto frontier. The identification of the true Pareto frontier without complete design space enumeration remains an unanswered question. However, in future cases, several different aspects of the MOGA may be altered to better aid in identification of the Pareto frontier. These include the removal of repeated designs and an increase in the mutation rate to promote design diversity, alteration of the termination criteria to require the hypervolume to remain constant for more generations, and a variable population size. Specific to the composite optimization

problem, the inclusion of an additional mutation phase in which two or more plies are selected and the orientation angles switched may promote diversity and increase the likelihood of the identification of the true Pareto frontier. This is an area for future work regarding composites, but does not apply to applications of this methodology to other problem sets.

Additionally, the initial convergence results using hypervolume should be run multiple times and the results aggregated and compared. After a single run, the Random Forest classifier is best-suited for use with composite panel optimization. However, when the convergence criteria was altered and the MOGA run multiple times, the Decision Tree classifier outperformed the others in terms of average function count and hypervolume. These results indicate that to identify the "best" classifier for use with composite panel optimization a larger set of results must be gathered.

REFERENCES

[1] K. Deb, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evoluationary Computation*, vol. 6, no. 2, pp. 182-197, 2002.

[2] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *IEEE International Conference on Neural Networks*, Perth, 1995.

[3] E. Aarts, J. Korst and W. Michiels, "Simulated Annealing," in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, New York, Springer Science+Business Media, Inc, 2005, pp. 187-210.

[4] A. Bekasiewicz, S. Koziel and W. Zieniutycz, "Design Space Reduction for Expedited Multi-Objective Design Optimization of Antennas in Highly Dimensional Spaces," in *Solving Computationally Expensive Engineering Problems*, Switzerland, Springer International Publishing, 2014, pp. 113-147.

[5] C. Bucher, "Adaptive Sampling- an Iterative Fast Monte Carlo Procedure," *Structural Safety*, vol. 5, no. 2, pp. 119-126, 1988.

[6] P. Backlund, D. Shahan and C. Seepersad, "Classifier-Guided Sampling for Discrete Variable, Discontinuous Design Space Exploration: Convergence and Computational Performance," *Engineering Optimization*, vol. 47, no. 5, pp. 579-600, 2014.

[7] K. Von Hagel, S. Joglekar, M. Pankow and S. Ferguson, "Exploring Robust Composite Panels Using Multi-Objective Optimization and Varying Load Conditions," in *15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Atlanta, 2014.

[8] H. Ghiasi, D. Pasini and L. Lessard, "Optimum Stacking Sequence Design of Composite Materials Part I: Constant Stiffness Design," *Composite Structures*, vol. 90, no. 1, pp. 1-11, 2009.

[9] G. Soremekun, Z. Gurdal, R. Haftka and L. Watson, "Composite Laminate Design Optimization by Genetic Algorithm with Generalized Elitist Selection," *Computers & Structures*, vol. 79, no. 2, pp. 131-143, 2001.

[10] R. Eberhart and J. Kennedy, "A New Optimizer Using Particle Swarm Theory," in *Sixth International Symposium on Micro Machine and Human Science* , Nagoya, 1995.

[11] D. Gorissen, I. Couckuyt, P. Demeester, T. Dhaene and K. Crombecq, "A Surrogate Modeling and Adaptive Sampling Toolbox for Computer Based Design," *The Journal of Machine Learning Research*, vol. 11, pp. 2051-2055, 2010.

[12] M. Tabatabaei, J. Hakanen, M. Hartikainen, K. Miettinen and K. Sindhya, "A Survey on Handling Computationally Expensive Multobjective Optimization Problems Using Surrogates: Non-nature Inspired Methods," *Structural and Multidisciplinary Optimization*, vol. 52, no. 1, pp. 1-25, 2015.

[13] D. Shahan and C. Seepersad, "Bayesian Network Classifiers for Set-Based Collaborative Design," *Journal of Mechanical Design*, vol. 134, no. 7, pp. 1-14, 2012.

[14] R. Malak and P. C.J.J, "Using Support Vector Machines to Formalize the Valid Input Domain of Predictive Models in Systems Design Problems," *Journal of Mechanical Design*, vol. 132, no. 10, pp. 1-14, 2010.

[15] P. Backlund and J. Eddy, "Autonomous Microgrid Design Using Classifier Guided Sampling," in *Proceedings of the ASME 2015 IDETC & CIE*, Boston, 2015.

[16] E. Galvan and R. Malak, "P3GA: An Algorithm for Technology Charaterization," *Journal of Mechanical Design*, vol. 137, no. 1, pp. 1-13, 2015.

[17] P. Backlund and J. Eddy, "Classifier-Guided Sampling for Complex Energy System Optimization," Sandia National Laboratories, Albuquerque, 2015.

[18] J. Han and M. Kamber, Data Mining: Concepts and Techniques, 2001.

[19] P. Hart, "The condensed nearest neighbor rule," *IEEE Transactions on Information Theory*, vol. 14, pp. 1966-1967, 1968.

[20] H. Zhang, "The Optimality of Naive Bayes," *AA*, vol. 1, no. 2, 2004.

[21] R. Quinlan, C4.5: programs for machine learning, 1992.

[22] L. Breiman, "Random Forest," *Machine Learning*, vol. 45, no. 5, pp. 1-35, 1999.

[23] M. K. Buckland and G. Fredric, "The relationship between recall and precision," *JASIS*, vol. 45, no. 1, pp. 12-19, 1994.