ASME 2017 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference
IDETC2017
August 6-9, 2017, Cleveland, Ohio

DETC2017-68332

BENCHMARKING THE PEFORMANCE OF A MACHINE LEARNING CLASSIFIER ENABLED MULTIOBJECTIVE GENETIC ALGORITHM ON SIX STANDARD TEST FUNCTIONS

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
Previous work tested a multi-objective genetic algorithm that was integrated with a machine learning classifier to reduce the number of objective function calls. Four machine learning classifiers and a baseline "No Classifier" option were evaluated. Using a machine learning classifier to create a hybrid multiobjective genetic algorithm reduced objective function calls by 75-85% depending on the classifier used. This work expands the analysis of algorithm performance by considering six standard benchmark problems from the literature. The problems are designed to test the ability of the algorithm to identify the Pareto frontier and maintain population diversity. Results indicate a tradeoff between the objectives of Pareto frontier identification and solution diversity. The "No Classifier" baseline multiobjective genetic algorithm produces the frontier with the closest proximity to the true frontier while a classifier option provides the greatest diversity when the number of generations is fixed. However, there is a significant reduction in computational expense as the number of objective function calls required is significantly reduced, highlighting the advantage of this hybrid approach.

INTRODUCTION
Objective function complexity, associated computation time and large design spaces provide the impetus for reducing the number of objective function evaluations required by modern optimization algorithms. To reduce computational cost, designers must often choose between using a low fidelity meta-model (reduced run time at the tradeoff of performance estimate accuracy) or using a high fidelity model (increased run time with greater accuracy in performance estimate). This tradeoff has resulted in the creation of various algorithms with the goal of reducing the number of function evaluations while still identifying optimal designs. Different approaches have resulted in optimization algorithms that include probability distributions [1], machine learning classifiers [2, 3, 4], and improved meta-models [5]. Each of these methods results in a reduced number of objective function evaluations and outperforms the standard heuristic optimization algorithm configurations for Particle Swarm Optimization [6], Simulated Annealing [7] and Genetic Algorithms [8] in multiobjective engineering design optimization problems.

The algorithm being benchmarked in this work was originally developed and proposed in [3]. A flowchart of the algorithm is shown in Figure 1. This algorithm combines a multi-objective genetic algorithm (MOGA) and a machine learning classifier. The algorithm follows the general process of a MOGA by starting with a random initial population, and proceeds through selection, crossover and mutation. After mutation, a classifier is trained using the previously evaluated designs where "good" designs are closer to the Pareto frontier and "bad" designs are farther from the Pareto frontier. The classifier is then used to label the child designs as "good" or "bad". Only "good" designs are evaluated, while "bad" children designs are discarded. Once the "good" child designs are evaluated they are added to the population, and the population is culled to maintain a consistent size. All evaluated designs are added to a repository that is used to retrain the classifier each generation. As the number of designs in the repository grows the classifier has more information for training purposes.

The algorithm in Figure 1 was tested in [3] using a composite panel optimization problem in which the load cases of compression and uniaxial tension were evaluated for a composite panel of 10 layers with orientation angles ranging from -90 to 90 degrees in increments of 10 degrees. Four different types of classifiers, and a baseline, no-classifier option were incorporated. Results from this multiobjective optimization problem demonstrated that this hybrid approach achieved a minimum reduction in objective function evaluations of 75% when compared to the no-classifier option, as shown in Table 1.

[1]Corresponding author

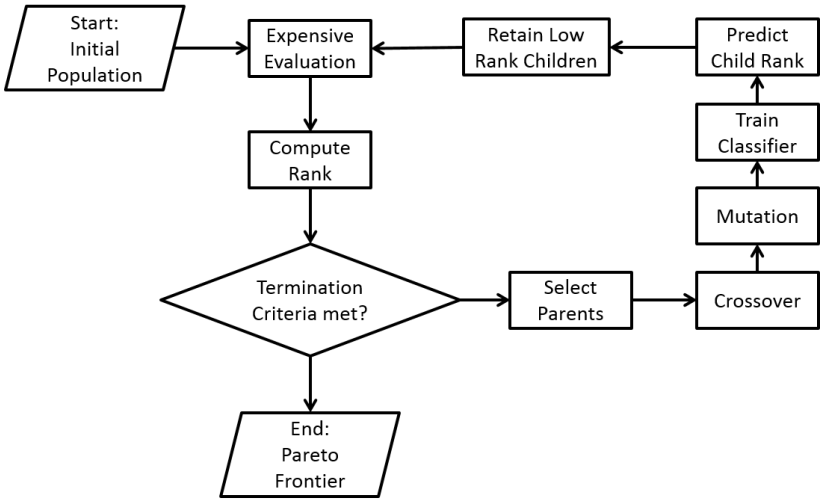

Figure 1: Algorithm for Benchmark Exploration [3]

Table 1: Summary of Evaluation Reduction when Compared to the No Classifier Option [3]

Classifier	Evaluation Reduction
Decision Tree	88%
k-Nearest Neighbor	85%
Naïve Bayes	75%
Random Forest	84%

The results from this problem were encouraging but only reflected the outcome of a single multiobjective optimization problem. The objective of this paper is to further assess performance of this approach across six standard benchmark problems from the literature. Using these benchmark problems, this work determines the problem type in which the following machine learning classifiers perform "best": k-Nearest Neighbor, Decision Tree, Naïve Bayes and Random Forest.

BACKGROUND

This work leverages various aspects of optimization and machine learning. The following subsections provide relevant information on the optimization and machine learning aspects of this work.

Function Reducing Optimization Methods

Evolutionary optimization algorithms use prior data to slowly move the optimal solution to the true global optimum. These algorithms generally require large numbers of expensive function evaluations and take considerable time to identify the set of non-dominated solutions.

Algorithms

Several optimization algorithms for minimizing objective function evaluations exist. Examples include Estimation of Distribution Algorithms (EDA) [1], Design Space Reduction [4], Surrogate-Based Optimization [5], and Classifier Guided Sampling (CGS) [2]. Each category of algorithm takes a slightly different approach to optimal solution identification when compared to the method described in [3].

EDAs build statistical models from designs of the previous iteration to identify promising candidate solutions [1]. There are no crossover or mutation operators like those that exist in typical evolutionary algorithms [9]. This is similar to the approach described in [3], as previous designs are used for predicting whether a design is close to the frontier. However, unlike EDAs, the approach implemented in [3] includes a crossover and mutation operator.

Design space reduction identifies ill-suited regions of the design space and removes those regions from consideration. This results in a restricted design space and allows the algorithm to identify optimal designs through fewer expensive evaluations [4]. In the method presented in [3], the classifiers perform the action of reducing the design space through evaluated designs. The classifiers learn which regions of the design space result in poor solutions and label designs in those regions as “bad”.

Surrogate-Based Optimization requires the development and testing of a meta-model to reduce the number of expensive evaluations [5, 10, 11]. The objective of the work in [3] is to avoid the use of a meta-model and instead reduce the number of expensive objective function evaluations.

In CGS, a weighted-sum single objective optimization problem is developed in which a Bayesian network classifier is used to predict whether child designs will improve the objective functions or not. Results indicate the classifier significantly reduces the number of objective function evaluations and converges upon the optima at a faster rate than genetic algorithms [2].

The work in [3] expands upon the concepts proposed in CGS through the addition of a second objective function and

completes a comparison of multiple classifiers. For the optimization problem used in [3], the Random Forest classifier completed the optimization in the fewest objective function calculations and performed better than the other classifiers.

These algorithms boast improved results over standard optimization algorithms. However, the method of reduction is highly problem dependent. The approach used for the removal of certain areas of the design space for one problem may not hold for a second unrelated problem. For example, CGS incorporates a Naïve Bayes classifier, but the results presented in [3] indicated that a Naïve Bayes classifier is not best suited for use in composite panel optimization.

Classification

Given the wide variety of available classification techniques, the correct learning algorithm for a specific application is not always apparent. In machine learning classification, the objective is to learn a concept from historical data, called training data, that can be used to assign previously unseen data points, or instances, to one of two or more categories, called classes [12]. It is believed that these types of models can be used to approximate the behavior of complex functions, and as a result, help isolate areas in design spaces that warrant further investigation by labeling new designs as "good" or "bad". Even though a specific classification model will likely not be a perfect facsimile of the actual problem, the idea is that the model may induce enough information about the underlying structure of the problem to isolate areas in the design space that deserve further investigation. In this work, classification models are explored from each of the following overarching families: instance-based classifiers, statistical classifiers, rule-based classifiers, and ensemble classifiers.

k-Nearest Neighbors (kNN)

Instance-based classifiers never truly build a model, but instead rely on their training data to classify new instances. In the case of the k-Nearest Neighbors (kNN) classifier [13], an unlabeled instance is classified according to the majority class found in the k training instances closest to its location in the feature space. Although instance based classifiers are simplistic, they do perform well when data points belonging to different classes are well separated in the feature space. However, it is important to use a distance metric that can uncover the separation between classes in order to obtain accurate classifications.

Naïve Bayes

Naïve Bayes is a statistical classification model that requires simple probability calculations to predict the class label of a new instance, and it has been shown to have good performance for a wide variety of applications [14]. A Naïve Bayes model makes the assumption that the effect of each attribute on the predicted class is independent of the other attributes [12]. This assumption simplifies the required calculations of the classifier, but is likely untrue in practice. Therefore, naïve Bayes performs best when an instance's attributes affect its class outcome with high levels of independence.

Decision Trees

Rule-based classifiers follow a series of "if-then" rules applied to the different attributes of an instance. Decision trees fit into this family because they are easily deconstructed into rules. Decision tree algorithms discover their tree in a top-down manner by choosing attributes one at a time and dividing the training instances into subsets according to the values of their attributes. The most important attribute is chosen as the top split node, the second most important attribute is considered at the next level, and so forth. For example, in the popular C4.5 algorithm [15], 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 have good predictive accuracy when the training data's attributes have a hierarchical structure in regard to determining class label. Still, decision trees are prone to overfitting [12].

Random Forest

Ensemble classifiers leverage multiple classification models to make a prediction. This is done by allowing each individual model to vote for what they believe should be the class label of the new instance, and then combing the votes according to some scheme. Random forests are a popular ensemble technique that creates many decision trees and uses a majority wins rule to classify a new instance. In random forest, decision trees are created using a random subset of the training data's attributes and instances [16]. It has been found that random forests can achieve very good accuracy, especially for tasks on which a single decision tree would already have good performance.

In review, the correct classifier to use for a particular application is not always apparent and depends on the structure of the underlying problem. Still, each classification model has its own strengths. kNN has good performance when the classes are separated in the design space. Naïve Bayes has good performance when there are high levels of independence between the problem's attributes. Decision Trees have good performance when classes can be segregated by creating top-down separations of the data. Finally, Random Forests can improve upon the performance of single-model classifiers. The objective of this paper is to identify problem characteristics contributing to the success or failure of a classifier.

METHODOLOGY

The approach taken in this work is outlined in Figure 2. The first step requires selecting an algorithm for evaluation and a problem with which to test. Problem set-up describes the process of setting or selecting the necessary options with the algorithm and problem choice. During the evaluation step, the algorithm is run. To adequately compare the results of the evaluation step, a set of metrics must be identified and calculated. The following subsections give greater detail into each of the steps of the methodology.

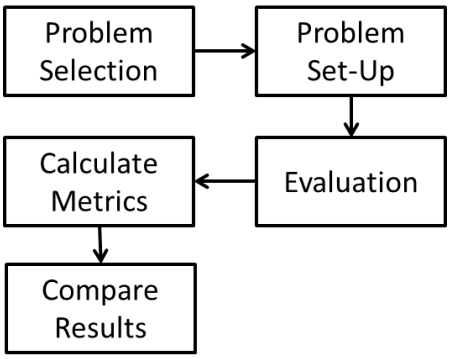

Figure 2: Comparison Methodology

Problem Selection

The algorithm analyzed in this approach is based off an NSGA-II [8] MOGA. This hybridized MOGA departs from traditional methods by using a classification model to identify which child designs are worthy of evaluation. The classifier is trained after each generation to incorporate knowledge uncovered from previously evaluated designs. For classifier training purposes, the non-domination rank value is set at 5. Meaning, when training each classifier, those previously evaluated designs with a rank of 5 or less are deemed "good" while those with a non-dominated rank greater than 5 are deemed "bad". After training the model and classifying the child designs, only child designs predicted to be "good" are evaluated. Further details regarding the approach used can be found in [3].

Problem Set-up

The six benchmark problems identified in [17] are used to compare five different MOGAs. Four of the MOGA cases incorporate one of the machine learning classifiers discussed in the previous section. Additionally, there is a "No Classifier" MOGA that is treated as a baseline. These problems highlight the known difficulties of genetic algorithms and other evolutionary algorithms to identify the Pareto frontier and maintain population diversity [8]. In standard form, each of the problems has the objectives given in Equation (1).

$$\begin{aligned} \text{Minimize:} \quad & T = (f_1(x), f_2(x)) \\ \text{Subject to:} \quad & f_2 = g(x) * h(x) \\ \text{where} \quad & x = (x_1, \dots, x_m) \end{aligned} \tag{1}$$

Evaluation

Each problem and classifier combination was run ten times and convergence was set at 100 generations. This maximum generation limit serves only as a means to terminate the algorithm after a set number of iterations. The No Classifier MOGA is also run for 100 generations and the goal is to compare solutions with respect to the true Pareto frontier for each benchmark problem. The population size was held constant at 100, arithmetic crossover is used with a crossover rate of 0.8 and the mutation rate is set at 0.1.

K-Nearest Neighbor, naïve Bayes, C4.5 decision tree and random forest classifiers were used with the objective of reducing the number of required function evaluations. More specifically, kNN was implemented with a Euclidean distance metric and one neighbor, that is k = 1. Random forests consisted of 100 trees each. The classifier training data attributes are the design variables associated with each of the benchmark problems. The class label corresponds to the proximity of the design with the current set of non-dominated results. That is, designs closer to the Pareto frontier are labeled "good" while those further from the frontier are labeled "bad."

Calculate Metrics

While many algorithms significantly reduce the number of expensive objective function algorithms. Little work exists comparing the algorithms and the function reducing, problem specific aspects of the algorithm. However, there exists much work comparing optimization algorithms without the attempts to reduce function evaluations through a machine learning classifier.

Baskar and Suganthan [18] compared a concurrent particle swarm optimization (CONPSO) algorithm with a standard particle swarm optimization algorithm through a series of six benchmark continuous optimization problems. The results indicated the CONPSO algorithm clearly outperformed the standard algorithm. Comparisons were conducted in terms of solution quality, average computation time and solution consistency.

In [19], a suite of 34 benchmarking problems are used to compare three different optimization algorithms. Performance is compared through the speed at which an algorithm reaches the optimum and the differential evolution algorithm is found to be the most robust.

In another optimization algorithm comparison, Zitzler et al compare eight different multi-objective algorithms on six test problems designed to highlight an algorithm's ability (or inability) to identify the Pareto frontier and maintain solution diversity. Results are compared by measuring the dominated area of the solution space, or hypervolume, and the percentage of one set of results that covers, or dominates, another [17]. These complimentary metrics of performance were presented in [20, 21].

The metrics provided in [17] are used for comparison of the identified Pareto frontiers for each of the 6 benchmark problems. Equation (2) gives the distance to the true Pareto frontier, $\bar{X}$ from an identified Pareto frontier, X'. To calculate M_1 the minimum distance between each point in the identified frontier and true frontier is calculated, summed and divided by the cardinality, or number of points in the identified set. Smaller values of M_1 are "better" as they indicate the identified frontier is closer in proximity to the true Pareto frontier.

$$M_1(X') = \frac{1}{|X'|}\sum_{a' \in X'} \min\{||a' - \bar{a}||: \bar{a} \in \bar{X}\} \quad (2)$$

Equation (3) considers the distribution and density of the identified Pareto frontier. The neighborhood parameter, σ indicates the number of design vectors in the identified neighborhood. The higher the value of M_2 the greater the distribution and density of identified Pareto points and therefore, the "better" the solution. Since the number of Pareto points influences the value for M_2 these values are normalized between 0 and 1 for better comparison.

$$M_2(X') = \frac{1}{|X'| - 1}\sum_{a' \in X'} |\{b' \in X'; ||a' - b'|| > \sigma\}| \quad (3)$$

Equation (4) measures the range of the identified Pareto frontier. In the case of two objectives, M_3 is the distance between the outermost two designs in the solution space.

$$M_3(X') = \sqrt{\sum_{i=1}^{m} \max\{||a_i' - b_i'||: a', b' \in X'\}} \quad (4)$$

Further analysis examines the precision and recall of the classifier for the final generation (trained classifier) of the algorithm. In this context, precision is the percentage of designs predicted by the classifier to be "bad" that are actually "bad". Recall is the percentage of actually "bad" designs that are predicted to be "bad" [22]. Through the analysis of the classifier and hypervolume, we extract those classifiers best suited for each of the benchmark problems.

Compare Results

Comparisons are conducted on a per benchmark test problem basis as problem characteristics influence the result and cross-problem comparison may be biased.

RESULTS FOR THE SIX BENCHMARK PROBLEMS

Each Test Problem was run ten times using the previously discussed classifiers and the No Classifier option. The results in the following subsections are the aggregation of each of the ten runs.

Using the values given in [17] for $g(x)$ and the ranges associated with x_1, a true frontier consisting of 100 Pareto points evenly spaced along f_1 was calculated for Test Problems 1-4 and 6. Due to the nature of Test Problem 5, the number of Pareto points of the true frontier has an upper limit of 31, so the entire frontier was enumerated for comparison.

Test Problem 1 – Convex Pareto frontier

The first Test Problem of the benchmark set has a convex Pareto frontier which is found when $g(x) = 1$. Equation (5) gives the necessary functions for minimization where $m = 30$ and $x_i \in [0,1]$.

$$\begin{aligned} f_1 &= x_1 \\ g(x_2, \dots, x_m) &= 1 + 9 * \sum_{i=2}^{m} \frac{x_i}{m-1} \\ h(f_1, g) &= 1 - \sqrt{f_1/g} \end{aligned} \quad (5)$$

After running the algorithm 10 times for each of the identified classifiers and the No Classifier option with a convergence criteria of 100 generations, the aggregate Pareto frontier was identified on a per classifier basis and is displayed in Figure 3. Given the defined termination criteria, none of the aggregate Pareto frontiers reached the true Pareto frontier.

The No Classifier frontier produces the lowest values consistent with the minimization objective. However, the no-classifier frontier spans less than half of the feasible solutions space for the frontier. This is also true of the Decision Tree, kNN and Random Forest frontiers. Only the naïve Bayes classifier produces a frontier that covers a significant range of the solution space.

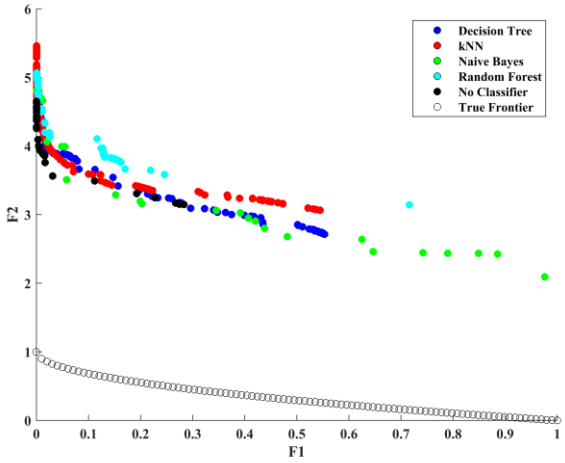

Figure 3: Test Problem 1 Aggregate Pareto Frontiers

Table 2 provides summary information for each of the aggregate Pareto frontiers displayed in Figure 3. The No Classifier frontier has the smallest extent and the naïve Bayes frontier the largest. From a proximity standpoint, the No Classifier frontier is closest to the true frontier. The number of designs in the frontier plays a role in the diversity measurement. Frontiers consisting of more Pareto points are likely to have larger values for diversity. Therefore, we disregard diversity for comparison.

Despite the differences in metrics, the identified aggregate Pareto frontiers in Figure 3 are comparable. That is, there exist trade-offs between each of the classifier frontiers and no frontier is superior to all others. The baseline, No Classifier MOGA required over 15,000 function evaluations to produce the frontier shown. Naïve Bayes, the frontier with the largest extent, is achieved in slightly less than 1,000 function

evaluations. This is a function evaluation savings of 93.4%, achieved because the children designs classified as being of poor quality are not evaluated. kNN, which produced the frontier with the greatest diversity measurement and most unique designs, saved 25.6% in function evaluations compared to the No Classifier MOGA and nearly tripled the next highest number of function evaluations required by Random Forest.

Table 2: Test Problem 1 Metrics for Performance Comparison

	Decision Tree	k-Nearest Neighbor	Naïve Bayes	Random Forest	No Classifier
Unique Designs	137	184	24	104	39
Hyper-Volume	6.00	6.34	8.21	9.14	3.33
Proximity M_1	2.99	3.27	2.35	3.57	3.07
Diversity M_2	15572	30586	522	7802	1256
Extent M_3	2.45	2.46	2.88	2.05	1.53
Function Count	2988	11445	1019	3639	15382
Function Call Percent Reduction	80.6%	25.6%	93.4%	76.3%	--

A secondary goal of this work is to evaluate the effectiveness of the classifier in terms of predictive ability. As previously discussed, this is done using precision and recall. Figure 4 shows the average precision for each classifier at each generation. All four classifiers start with relatively high precision values, but experience a severe reduction in precision after a few generations. The low average precision values indicate an inability of the classifier to accurately differentiate "good" designs from "bad".

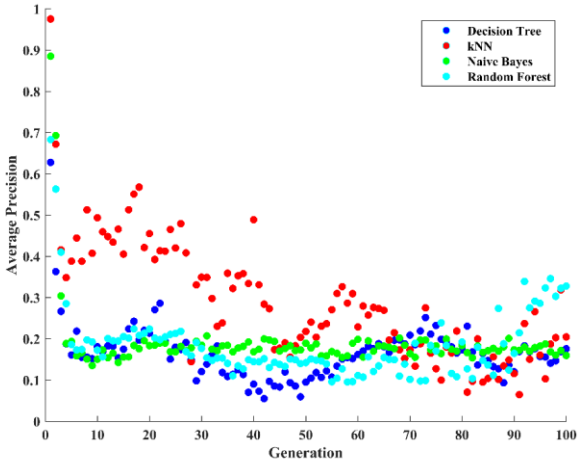

Figure 4: TP 1 Average Classifier Precision per Generation

Similarly, Figure 5 displays, for each generation and classifier, the average percentage of actually "bad" designs that are predicted to be "bad". All classifiers experience an increase in recall in the early generations of the algorithm. However, kNN, Decision Tree and Random Forest decrease in average recall as the algorithm reaches the stopping criteria. Naïve Bayes reaches and maintains nearly perfect recall around generation 30 until the convergence criteria of 100 generations.

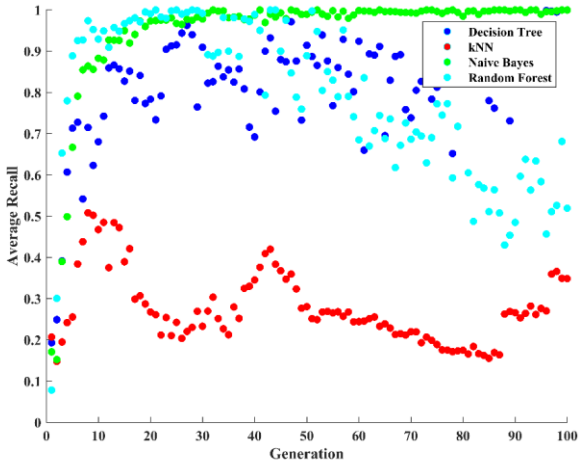

Figure 5: TP 1 Average Classifier Recall per Generation

Test Problem 2 – Nonconvex Pareto frontier

Opposite to Test Problem 1, the second Test Problem has a nonconvex frontier also found when $g(x) = 1$. The specifics for Test Problem 2 are given in Eq. (6) where $m = 30$ and $x_i \in [0.1]$.

$$\begin{aligned} f_1 &= x_1 \\ g(x_2, \ldots, x_m) &= 1 + 9 * \sum_{i=2}^{m} \frac{x_i}{m-1} \\ h(f_1, g) &= 1 - \left(\frac{f_1}{g}\right)^2 \end{aligned} \tag{6}$$

The aggregate non-dominated solutions from the combination of runs are shown in Figure 6. The aggregate Pareto frontiers favor lower values for F_1 and higher values for F_2. As a result the generated frontiers are sparse as the values for F_1 increase. Further, the aggregate Pareto frontiers do not fully capture the nonconvex behavior of the true Pareto frontier. Instead, the aggregate frontiers appear closer to a straight line than convex. However, this is a scale problem and slight nonconvex behavior is apparent when the scale for F_2 is altered.

Once again, the No Classifier option Pareto frontier dominates those frontiers of the other classifiers. As with Test Problem 1, none of the algorithms reach the actual Pareto frontier given the convergence criteria specified. Future work will change the convergence criteria so that the number of generations is increased, and a comparison will also be completed when the number of function calls used is held constant.

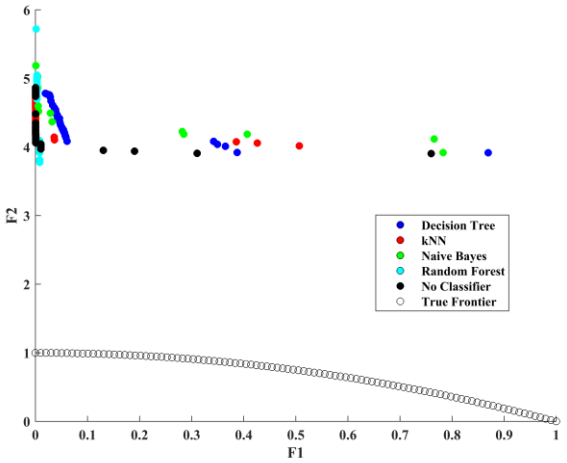

Figure 6: Test Problem 2 Pareto Frontiers

To better compare the results of Figure 6, the results displayed in Table 3 describe the proximity of the identified frontier to the true Pareto frontier, and the diversity and extent of the identified frontier. In pursuit of function count reduction, the naïve Bayes classifier once again provides the greatest reduction in function calls with an average of over 95% savings. The naïve Bayes classifier also produces the Pareto frontier with the greatest extent, nearly doubling that of the No Classifier MOGA. Unfortunately, in the context of unique designs, hypervolume, proximity and diversity, the naïve Bayes classifier underperforms the other classifier options.

Table 3: Test Problem 2 Metrics for Performance Comparison

	Decision Tree	k-Nearest Neighbor	Naïve Bayes	Random Forest	No Classifier
Unique Designs	37	87	12	60	82
Hyper-Volume	14.09	8.45	13.93	0.16	11.70
Proximity M_1	3.39	3.38	3.49	3.67	3.44
Diversity M_2	1082	5664	126	2482	4532
Extent M_3	1.23	0.99	1.49	1.94	1.23
Function Count	2184	12120	707	3322	15389
Function Call Percent Reduction	85.8%	21.2%	95.4%	78.4%	--

In consideration of the "best" performing classifier in terms of precision and recall, Figure 7 displays the average precision of each classifier for each of the 100 generations. Classifier precision reduces severely in the first 10 generations. Initially, all classifiers boast average precision values of nearly 100%, but as the generations continue, the precision appears to plateau between 0.2 and 0.4, depending on the classifier. Random Forest experiences an even larger decline in average precision in the final generations of the algorithm. kNN precision values suffer greatly after generation 40. These low precision values indicate that the classifiers are predicting several "good" designs as "bad" and this is likely responsible for the lack of nonconvex behavior found in the Pareto frontiers in Figure 6.

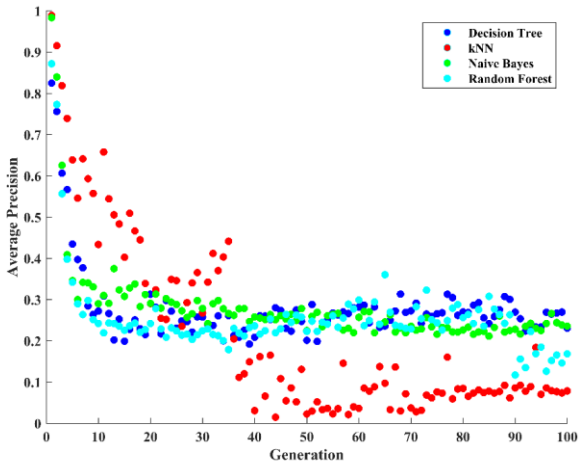

Figure 7: TP 2 Classifier Average Precision per Generation

Further analysis of the classifiers for Test Problem 2 produced the recall results presented in Figure 8. Despite the inability of the classifiers to accurately differentiate between "good" and "bad" designs, the classifiers do appropriately classify most "bad" designs as "bad" with the exception of kNN. Unfortunately, the high recall value is likely a result of the classifiers predicting all designs as "bad". In combination with the poor average precision values, the kNN classifier also produces poor average recall values. The combination of these low values is likely the cause of the large number of expensive function evaluations.

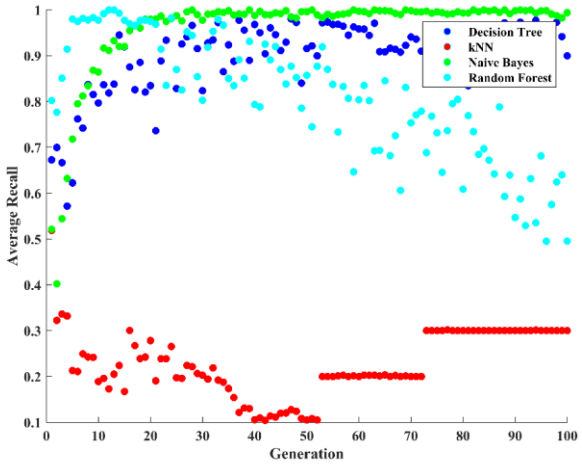

Figure 8: TP 2 Classifier Average Recall per Generation

Test Problem 3 – Discrete solution space

Test Problem 3 seeks to examine the ability of an optimization algorithm to handle a discrete solution space as the Pareto frontier consists of several distinct convex pieces. Equation (7) shows the necessary components of the minimization problem where $m = 30$ and $x_i \in [0,1]$.

$$f_1 = x_1$$
$$g(x_2, \ldots, x_m) = 1 + 9 * \sum_{i=2}^{m} \frac{x_i}{m-1} \quad (7)$$
$$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} - \left(\frac{f_1}{g}\right) \sin(10\pi f_1)$$

The aggregate Pareto frontiers for Test Problem 3 are shown in Figure 9. Once again, the No Classifier frontier is concentrated toward the smaller values for F_1 and does not expand into the larger function values. On the other hand, the naïve Bayes frontier spans the same length of the solution space as the true frontier although the values for F_2 are larger. The lack of convergence to the true frontier is a result of the prematurely enforced convergence of 100 generations.

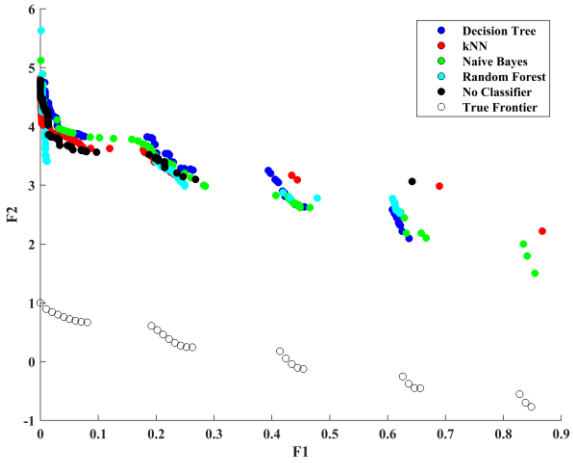

Figure 9: Test Problem 3 Aggregate Pareto Frontiers

Further investigation into the frontiers in Figure 9 results in the metrics presented in Table 4. As with the previous test problems, naïve Bayes produces the frontier with the greatest extent and the fewest number of function evaluations with a savings of 93%. Additionally, the naïve Bayes frontier has the smallest proximity value and is therefore, closest to the true frontier. However, in comparison with the other classifier options, naïve Bayes is outperformed by Decision Tree and Random Forest in terms of hypervolume and all other options in terms of diversity. From the results in Table 4, no classifier is clearly superior to all other classifiers.

Table 4: Test Problem 3 Metrics for Performance Comparison

	Decision Tree	k-Nearest Neighbor	Naïve Bayes	Random Forest	No Classifier
Unique Designs	116	170	91	128	106
Hyper-Volume	6.90	8.67	7.39	5.89	6.93
Proximity (M_1)	2.67	3.04	2.20	2.54	3.06
Diversity (M_2)	12068	25270	6846	14434	9706
Extent (M_3)	2.72	2.75	3.72	3.17	1.84
Function Count	3755	13802	1086	4723	15402
Function Call Percent Reduction	75.6%	10.4%	93%	69.3%	--

To explore the precision and recall values for Test Problem 3, Figure 10 displays the average precision for each of the classifiers for each of the 100 generations. As is consistent across Test Problems 1 and 2, the average precision for Test Problem 3 severely decreases in the early generations. Naïve Bayes holds somewhat steady at 0.4 across all generations while Decision Tree, kNN and Random Forest continue to decline. The mostly constant average precision for naïve Bayes may explain the large computational savings seen in Table 4.

The Decision Tree and Random Forest classifiers display similar behavior in that there is a steady decrease in precision as the number of generations increase and then the average values begin to increase with the larger generations. This similar behavior is likely a result of the Decision Tree as a Random Forest consists of several Decision Trees and a voting mechanism. This indicates that Decision Trees are not well suited for problems with trigonometric functions.

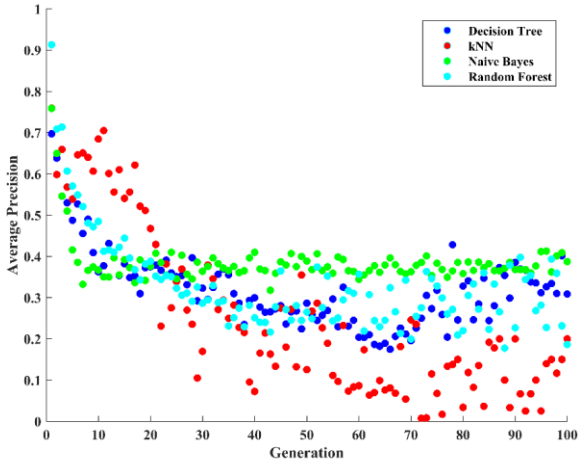

Figure 10: TP 3 Classifier Average Precision per Generation

In a similar manner, Figure 11 shows the average recall for each generation for each classifier. While naïve Bayes does not initially perform "best", naïve Bayes continually improves its average recall before plateauing at nearly perfect recall. On the other hand, Random Forest quickly provides the highest average recall values, but continually decreases in average recall as generations increase. Similarly, the average recall values for Decision Tree decline as the number of generations increases. kNN appears almost piecewise in terms of recall values. The nearly perfect prediction of "bad" designs as "bad" for the naïve Bayes classifier likely plays a role in the large reduction in expensive function evaluations. This means the classifier is ensuring no "bad" designs are evaluated.

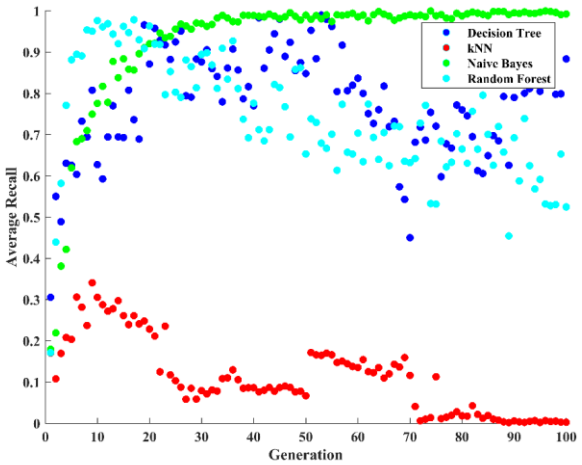

Figure 11: TP 3 Classifier Average Precision per Generation

Test Problem 4 - Multimodality

The fourth Test Problem tests the ability of the algorithms to handle multimodality as there exist 21^9 local Pareto frontiers. The global Pareto frontier occurs when $g(x) = 1$. Equation (8) shows the functions for f_1, g and h. In Eq. (8), $m = 10$ while $x_1 \in [0,1]$ and $x_2, \dots, x_m \in [-5,5]$.

$$\begin{gathered} f_1 = x_1 \\ g(x_2, \dots, x_m) = 1 + 10(m-1) + \sum_{i=2}^{m} (x_i^2 - 10\cos(4\pi x_i)) \\ h(f_1, g) = 1 - \sqrt{f_1/g} \end{gathered} \tag{8}$$

Figure 12 shows the aggregate Pareto frontiers identified by each of the classifiers options and the No Classifier MOGA. Clearly, the algorithm, including the baseline MOGA, struggles with the multimodality aspect of the test problem. The aggregate Pareto frontiers do not mimic the behavior of the true frontier and are concentrated around exceptionally small values for F_1 and extremely large values for F_2.

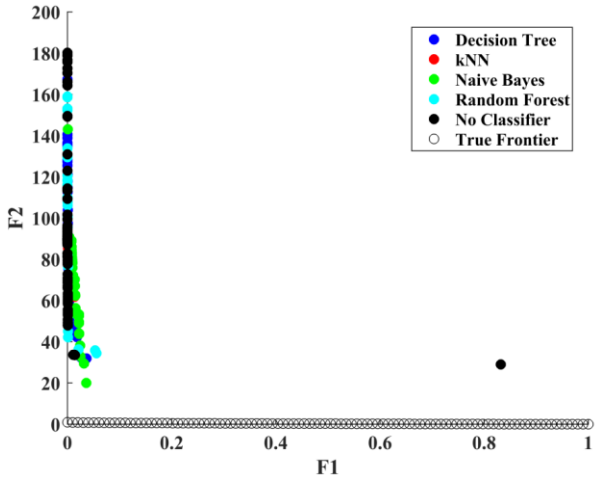

Figure 12: Test Problem 4 Aggregate Pareto Frontiers

From Figure 12, it is difficult to ascertain which Pareto frontiers are "better" as they clearly do not mimic the behavior of the true frontier. Table 5 summarizes metrics for each of the aggregate Pareto frontiers. The naïve Bayes classifier produces the greatest reduction in function calls with over 83% and is comparable to the No Classifier MOGA for the metrics of proximity and extent. However, the No Classifier frontier consists of several more unique designs and boasts a much larger hypervolume. The large hypervolume calculation is a result of the single Pareto point in Figure 12 with a value for F_1 greater than 0.2.

Table 5: Test Problem 4 Metrics for Performance Comparison

	Decision Tree	k-Nearest Neighbor	Naïve Bayes	Random Forest	No Classifier
Unique Designs	68	5	35	27	169
Hyper-Volume	76.96	53.16	144.29	87.40	822.73
Proximity M_1	80.23	68.47	66.23	98.92	70.07
Diversity M_2	4110	20	1186	702	25772
Extent M_3	135.4	23.34	123.01	124.50	151.22
Function Count	5571	12805	2593	6910	15408
Function Call Percent Reduction	63.8%	16.9%	83.2%	55.2%	--

Despite the poorly identified Pareto frontiers displayed in Figure 12, the classifiers performed well in terms of average precision as shown in Figure 13. Unlike previous test problems, the average precision values increase after the first few generations, except for naïve Bayes. In terms of consistency, however, naïve Bayes appears to provide the most consistent

average values for precision while the remaining classifiers fluctuate between high and low average precision values.

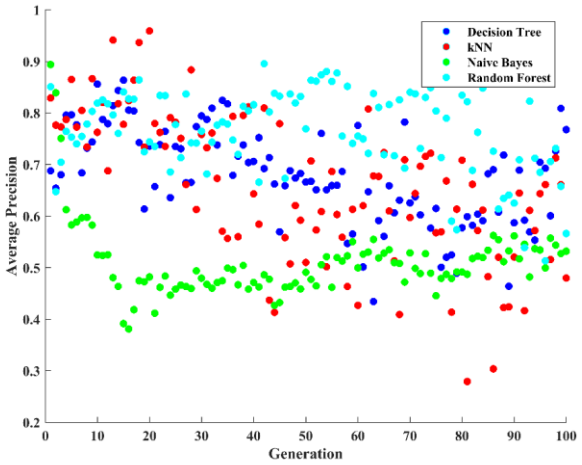

Figure 13: TP 4 Classifier Average Precision per Generation

Similarly, the average classifier recall for each generation is displayed in Figure 14. The average recall values have nearly inverse behavior of the average precision values. For example, naïve Bayes consistently has the lowest average precision values and the highest average recall values. kNN has the highest average precision values and the lowest average recall values. This inverse behavior implies the classifier does not identify many "bad" designs, but when it does, it accurately classifies them.

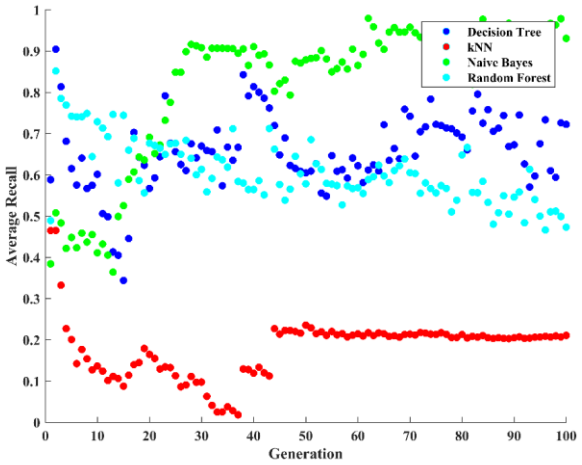

Figure 14: TP 4 Classifier Average Recall per Generation

Test Problem 5 – Binary design string

Unlike the other test problems, in Test Problem 5 each design variable is a binary string. As a result, the crossover method for Test Problem 5 is k-point crossover with k = 3. Equation (9) gives the necessary sub-functions for the minimization problem where $m = 11$ and $x_i \in [0,1]$. The first design variable, x_1 is a binary string of length 30 while the remaining variables are of length 5. The Pareto frontier is formed when $g(x) = 10$.

$$f_1(x_1) = 1 + u(x_1)$$

$$g(x_2, \ldots, x_m) = \sum_{i=2}^{m} v\big(u(x_i)\big)$$

$$h(f_1, g) = 1/f_1 \tag{9}$$

where $u(x_i)$ gives the number of ones in x_i and

$$v\big(u(x_i)\big) = \begin{Bmatrix} 2 + u(x_i) & if\ u(x_i) < 5 \\ 1 & if\ u(x_1) = 5 \end{Bmatrix}$$

Figure 15 display the aggregate Pareto frontiers identified for each classifier and the no classifier baseline option. Visually, the No Classifier option appears to outperform the other classifiers in terms of proximity to the true frontier. However, in terms of density, the naïve Bayes and Random Forest classifiers produce the more diverse and dense Pareto frontiers.

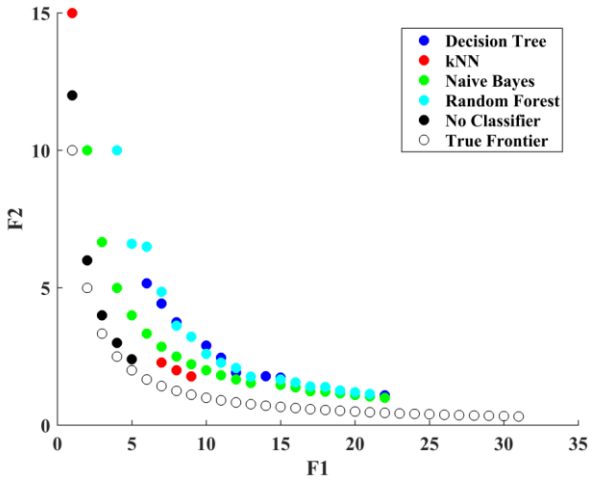

Figure 15: Test Problem 5 Aggregate Pareto Frontiers

Table 6 provides the metrics for comparison for the aggregate Pareto frontiers in Figure 15. The Decision Tree aggregate Pareto frontier is accomplished with a reduction in expensive function evaluations over 95%. However, this reduction in function evaluations comes at the expense of frontier extent, hypervolume and proximity. kNN provides a large diversity metric and the smallest proximity measurement, but provides very little in computational savings at less than 10%.

Table 6: Test Problem 5 Metrics for Performance Comparison

	Decision Tree	k-Nearest Neighbor	Naïve Bayes	Random Forest	No Classifier
Unique Designs	16	143	261	31	121
Hyper-Volume	126.27	167.30	203.86	249.78	127.2
Proximity M_1	1.81	0.63	1.38	1.68	0.72
Diversity M_2	228	12056	55458	880	7214
Extent M_3	16.51	15.45	21.93	19.17	10.4
Function Count	429	9014	2289	756	9899
Function Call Percent Reduction	95.7%	8.9%	76.9%	92.4%	--

To explore the average precision of each classifier for Test Problem 5, Figure 16 displays the average precision per generation. The precision values experience a large decline in the first few generation of the algorithm, but reach a steady state after approximately generation 20. Decision Tree and Random forest provide the highest levels of average precision with Decision Tree slightly outperforming Random Forest. Naïve Bayes continues to decline in average precision as the generations continue. kNN performs extremely poorly in the later generations with precision values of nearly zero. This means kNN classifies nearly every design as "good".

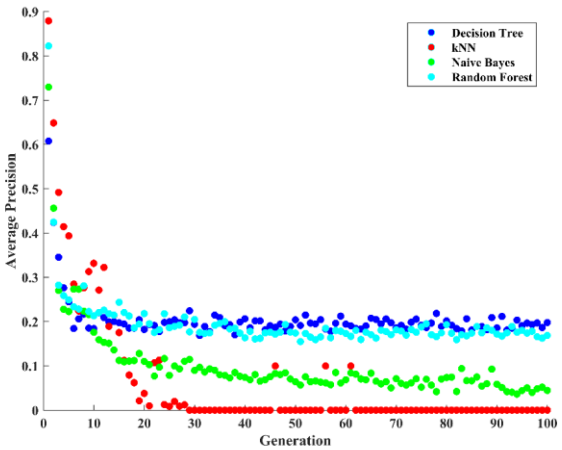

Figure 16: TP 5 Classifier Average Precision per Generation

Along with the consistent average precision values for Decision Tree and Random Forest, the classifiers also provide excellent average recall values as can be seen in Figure 17. While kNN struggles with consistency and eventually produces recall values of zero, Decision Tree and Random Forest reach average recall values of nearly 1 meaning nearly all actually "bad" designs are predicted to be "bad."

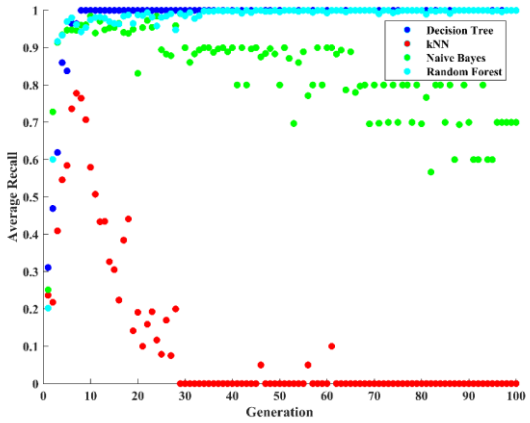

Figure 17: TP 5 Classifier Average Recall per Generation

Test Problem 6 – Non-uniformly distributed frontier

In Test Problem 6, the ability of the algorithm to identify non-uniformly distributed designs along the Pareto frontier and low solution density near the global Pareto frontier is evaluated. Equation (10) displays relevant equations for Test Problem 6 where $m = 10$ and $x_i \in [0,1]$. The Pareto frontier is found when $g(x) = 1$.

$$f(x_1) = 10 - \exp(-4x_1)\sin^6(6\pi x_1)$$

$$g(x_2, \dots, x_m) = 1 + 9 * ((\sum_{i=2}^{m} x_i)/(m-1))^{0.25} \qquad (10)$$

$$h(f_1, g) = 1 - \left(\frac{f_1}{g}\right)^2$$

Figure 18 displays the aggregate Pareto frontiers for Test Problem 6. The algorithm struggles to identify Pareto frontiers and for the majority of the classifiers, the identified Pareto frontier consists of only a single Pareto point. As can be seen from the true frontier, the density of the Pareto frontier increases as the value for F_1 increases. The implemented algorithms attempt to capture this behavior, but fail to identify any data points for values of F_1 less than 1.

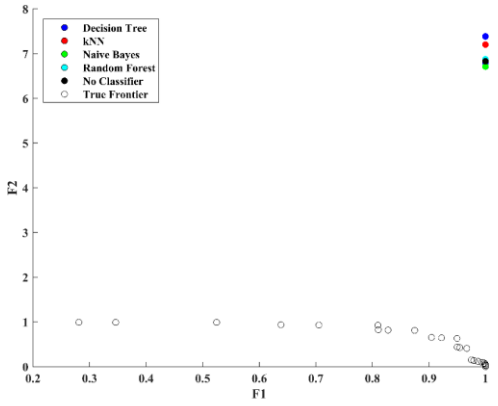

Figure 18: Test Problem 6 Aggregate Pareto Frontiers

To more accurately compare the frontiers identified in Figure 18, Table 7 displays the performance metrics for each of the classifiers and the No Classifier option. The only classifier to produce an actual frontier is Decision Tree. Therefore, a diversity and extent measurement can only be calculated for Decision Tree. In terms of function count, the naïve Bayes classifier produces a better data point in terms of hypervolume and proximity than the no classifier option in over 97% fewer function evaluations.

Table 7: Test Problem 6 Metrics for Performance Comparison

	Decision Tree	k-Nearest Neighbor	Naïve Bayes	Random Forest	No Classifier
Unique Designs	3	1	1	1	1
Hyper-Volume	7.39	7.20	6.71	6.87	6.83
Proximity M_1	5.99	6.23	5.74	5.90	5.86
Diversity M_2	4	0	0	0	0
Extent M_3	0.67	0	0	0	0
Function Count	2253	11534	454	4338	15404
Function Call Percent Reduction	85.4%	25.1%	97.1 %	71.8%	--

To further explore the results for Test Problem 6, Figure 19 displays the average precision for each classifier. An extreme reduction in average precision occurs for all classifiers in the early generations. However, as the generations increase, precision for Decision Tree, naïve Bayes and Random Forest levels out. kNN, on the other hand, appears sporadic in its average precision and produces values ranging from slightly more than 0.1 to nearly 1 across all the generations.

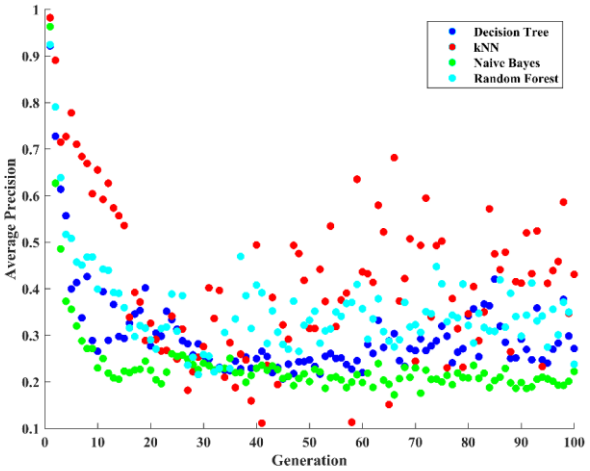

Figure 19: TP 6 Average Classifier Precision per Generation

Average recall, the percentage of actually "bad" designs that are predicted to be "bad", is displayed in Figure 20. The kNN classifier is outperformed by all other classifiers. Decision Tree experiences a continual climb in average recall while Random Forest experiences a continual decline. Naïve Bayes, however, reaches nearly perfect average recall in relatively short order and maintains it throughout the course of the algorithm.

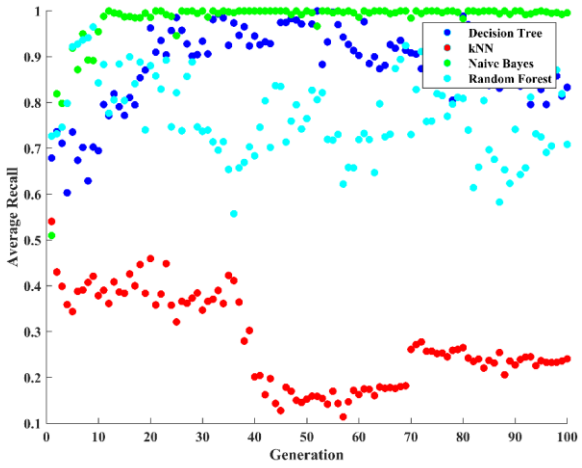

Figure 20: TP 6 Average Classifier Recall per Generation

DISCUSSION

The primary goal of this work is the incorporation of a machine learning classifier into a MOGA to reduce the overall number of function calls required while still producing results comparable to the no classifier, baseline implementation. A secondary goal of the work is the exploration of the classifier results and recommendations associated with classifier selection. The following subsections independently discusses these objectives in more detail.

Optimization Results

A theme across all of the test problems evaluated is the tradeoff between a low proximity value and a high diversity value. That is, in no test problem could a "best" solution be identified. A sacrifice must occur whether it is solution diversity or an increased distance from the true frontier. When the number of objective function calls is added to the discussion, the classifier option always becomes "better" than the No Classifier option. However, when given the choice between two classifier options, the designer must choose proximity, diversity or objective function count when convergence is based solely on the number of generations processed. In the evaluated test problems, the computation time for each objective function is trivial. However, for more complex engineering problems, the function count reduction may prove to be the most important metric for classifier choice.

The classifier options appear to perform well for Test Problems 1 and 2. Although, visually, there appears to be a decrease in solution extent and diversity in Test Problem 2.

The discrete solution space in Test Problem 3 poses little difficulty for the classifier options as their proximity and diversity metrics are superior to the No Classifier option. However, it appears that the identified classifier frontiers are denser for smaller values of F_1 and larger values of F_2. This indicates that each classifier is robust to a discrete solution space, with the Decision Tree, Random Forest and Naïve Bayes classifiers better able to traverse the solution space.

All classifiers and the No Classifier option struggle with the multimodality present in Test Problem 4. In general, the solutions are able to move past the local Pareto frontiers toward the true frontier. However, this movement is at the cost of solution diversity. These results support the overall conclusion that there is a tradeoff between proximity and diversity.

The binary strings present in Test Problem 5 pose little difficulty to the classifiers and No Classifier option. This is not entirely surprising as the number of design combinations for Test Problem 5 is much smaller than that of the other test problems.

Similar to the results of Test Problem 4, the results of Test Problem 6 indicate an inability of the algorithm to provide proximity and diversity. It is possible that the classifiers struggle with the non-uniformity of the solution space, but this is unlikely to be the only contributing factor as the No Classifier option also struggled. The low solution density near the Pareto frontier may also play a role. To determine the contributing factor, test problems to evaluate non-uniformity and low solution density must be developed and run independently. Across all six test problems, the No Classifier option performs poorly in terms of population diversity. This may be a result of the ability of the classifier to more accurately remove "bad" regions of the design space. It is also possible that the classifiers help to prevent the "rabbit-hole" behavior present by the No Classifier option for Test Problems 4 and 6.

Since the No Classifier option requires more than double the objective function evaluations with the exception of kNN, it is removed from discussion as a viable solution. This holds as the goal of the algorithm is to reduce objective function calls Table 8 displays the "best" choice for each test problem and metric.

In Table 8, the classifiers that provide the lowest proximity value, greatest diversity value, greatest extent value and smallest function count on a per test problem basis are identified. The choice of a "winner" is done by simple majority rules. In this way, the choice of classifiers is down-selected to Decision Tree and Naïve Bayes. While these two classifiers may not provide the "best" values for each of the calculated metrics, their selection for a given problem may lead to an overall "better" solution than the choice of another classifier.

Classifier Results

Each data point in the precision and recall per generation plots is the average precision or recall value for a particular classifier and generation across all 10 runs of the algorithm. In some instances, the classifier did not predict any "bad" designs or no "bad" designs existed. In these instances, the precision and/or recall value is undefined. Therefore, these instances are removed from the data set. Table 9 shows the number of occurrences of an undefined precision or recall value for a given problem and classifier combination.

From Table 9, kNN most frequently results in an undefined precision or recall value. In opposition, only for Test Problem 5 did naïve Bayes produces any undefined precision or recall values. The large number of undefined precision values for kNN play a role in the large number of expensive function evaluations as the classifier is frequently classifying all designs as "good" and must therefore, evaluation them with the expensive function.

Table 8: Classifier Results Comparison

	Test Problem 1	**Test Problem 2**	**Test Problem 3**	**Test Problem 4**	**Test Problem 5**	**Test Problem 6**	**Winner**
Proximity M_1	Naïve Bayes	Random Forest	Naïve Bayes	Random Forest	k-Nearest Neighbor	Naïve Bayes	Naïve Bayes
Diversity M_2	k-Nearest Neighbor	Decision Tree	k-Nearest Neighbor	Decision Tree	Naïve Bayes	Decision Tree	Decision Tree
Extent M_3	Naïve Bayes	Naïve Bayes	Naïve Bayes	Decision Tree	Naïve Bayes	Decision Tree	Naïve Bayes
Function Count	Naïve Bayes	Naïve Bayes	Naïve Bayes	Naïve Bayes	Decision Tree	Naïve Bayes	Naïve Bayes
Winner	Naïve Bayes	Naïve Bayes	Naive Bayes	Decision Tree	Naïve Bayes	Decision Tree/Naïve Bayes	

Table 9: Undefined Precision and Recall Occurrence per 1000 Generations

Problem	Classifier	Undefined Precision	Undefined Recall
Test Problem 1	Decision Tree	6	31
	k-Nearest Neighbor	306	2
	Naïve Bayes	0	0
	Random Forest	27	20
Test Problem 2	Decision Tree	3	1
	k-Nearest Neighbor	427	0
	Naïve Bayes	0	0
	Random Forest	8	17
Test Problem 3	Decision Tree	35	5
	k-Nearest Neighbor	499	7
	Naïve Bayes	0	0
	Random Forest	33	13
Test Problem 4	Decision Tree	48	0
	k-Nearest Neighbor	196	3
	Naïve Bayes	0	0
	Random Forest	49	0
Test Problem 5	Decision Tree	0	0
	k-Nearest Neighbor	648	553
	Naïve Bayes	62	140
	Random Forest	0	0
Test Problem 6	Decision Tree	18	3
	k-Nearest Neighbor	123	2
	Naïve Bayes	0	0
	Random Forest	7	10

To better explore the classifiers and their average precision values, Figure 21 shows the average precision values for each classifier differentiated by test problem on a single plot. From the figure, all four classifiers produce the highest average precision values for Test Problem 4. Recall, Test Problem 4 is designed to evaluate the ability of the algorithm to handle multimodality. Overall, the classifiers and baseline algorithm produced disappointing Pareto frontiers when compared to the true frontier. One potential reason for the higher level of precision across all four classifiers may be the relatively small number of design variables in Test Problem 4. However, Test Problem 6 also has only 10 design variables and no obvious pattern across classifiers can be extracted.

The average precision value for each classifier and problem combination experiences a reduction in average precision in the early generations of the algorithm. In some instances the average precision recovers from the initial reduction. In the early generations, the amount of training data is relatively small and there is a more equal balance between "good" and "bad" designs. However, as the number of generations increases, the training data becomes skewed and the number of "good" designs decreases while the number of "bad" designs increases. Therefore, the classifiers tend to err on the side of classifying designs as "bad". One potential solution to this problem is alteration of the cost matrix. That is, adjusting the cost matrix can make the misclassification of a "good" design more expensive, helping alleviate the challenges associated with these imbalanced data sets.

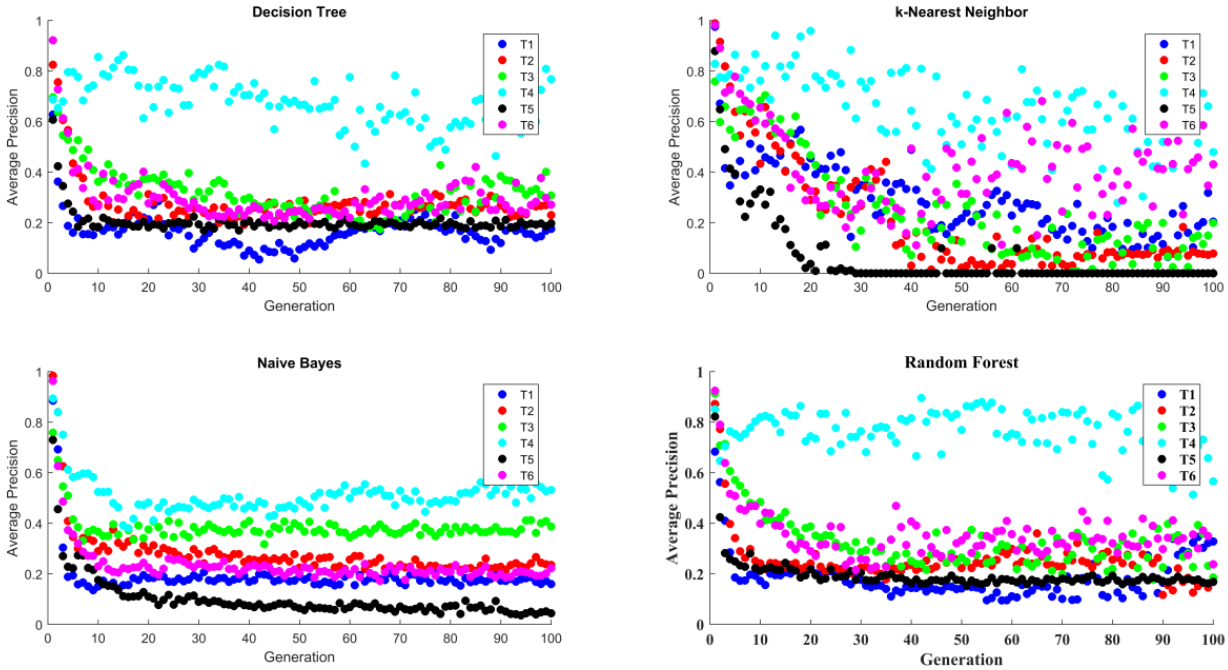

Figure 21: Average Precision by Classifier

Figure 22 displays the average recall per generation and test problem for each of the classifiers implemented. The kNN classifier consistently produces lower average recall values for all of the test problems than the other classifiers. This means kNN labeled large amounts of "bad" designs as good. As mentioned in previous sections, the low average recall values likely resulted in the large number of expensive function evaluations.

With the exception of Test Problem 5, Random Forest and Decision Tree experience variability in their respective average recall values. Random Forest appears to peak in the earlier generations and continually declines as the algorithm progresses. Decision Tree has slightly more variability but does not have the overall downward trend apparent for Random Forest. Both Decision Tree and Random Forest achieve nearly perfect average recall for Test Problem 5 in early generations and maintain it until algorithm convergence. One potential explanation for this is the reduced design space associated with Test Problem 5. There are 80 binary values in a design string for Test Problem 5 which translates to 2^{80} design combinations. However, for the other test problems there are countably infinite values in the range [0.1] for each of the 10 or 30 design variables depending on the test problem.

Naïve Bayes boasted the most consistent average precision values for all test problems and behaves similarly when considering average recall. Unlike Decision Tree and Random Forest, naïve Bayes struggles with Test Problem 5. The average recall value reaches a maximum in the early generations and systematically declines as the algorithm progresses. Naïve Bayes assumes independence between variables and perhaps this assumption is not valid for Test Problem 5.

It is also worth noting that Test Problem 4, which consistently had the highest average precision values, has consistently lower average recall values. The continual expensive evaluation of "bad" designs for Test Problem 4 may have contributed to the poor Pareto frontiers identified for Test Problem 4. As further evidence, the Pareto frontiers (or points) identified for Test Problem 6 are less than stellar and the average precision and recall for Test Problem 6 are relatively low with exceptions for kNN for precision and naïve Bayes for recall.

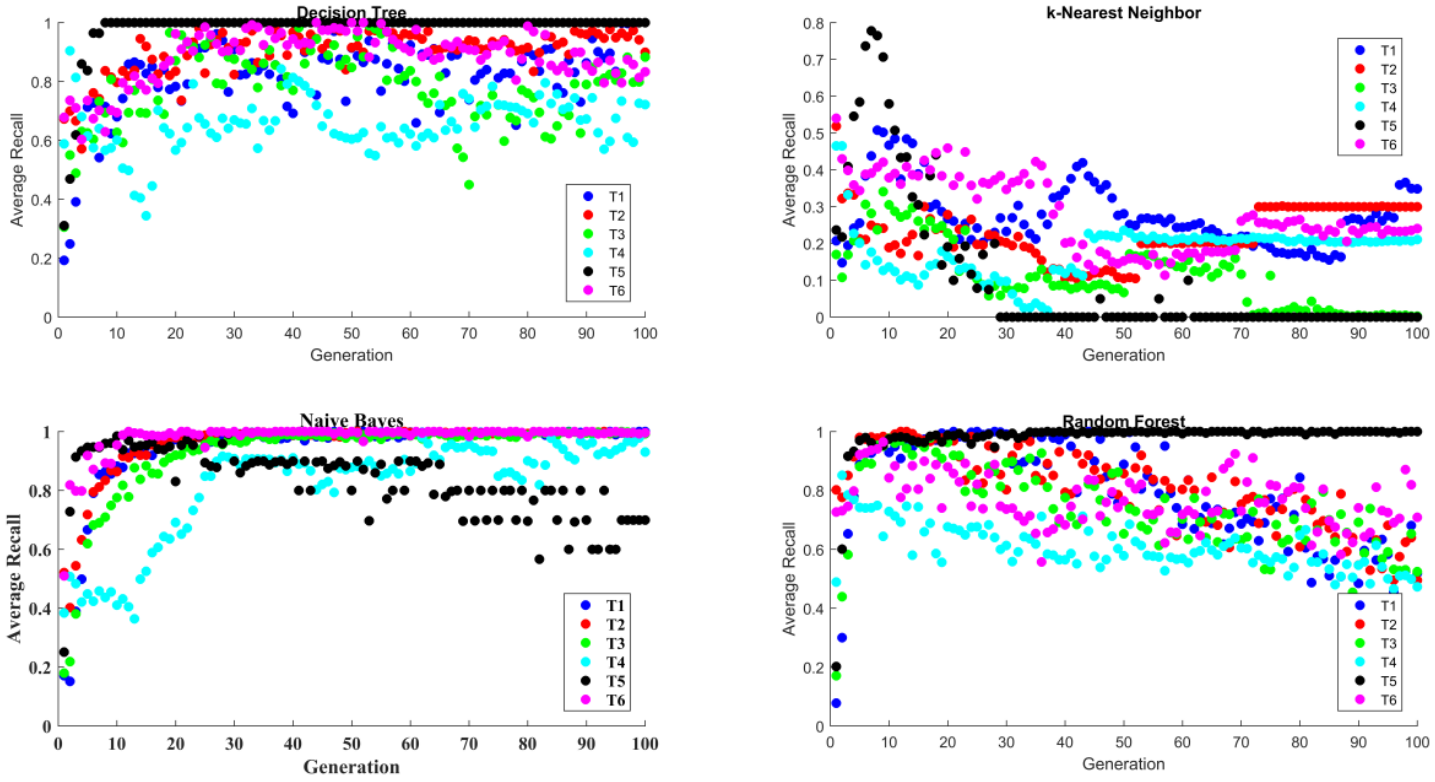

Figure 22: Average Recall by Classifier

CONCLUSION AND FUTURE WORK

The problems used in this work seek to test the ability of an algorithm to identify the Pareto frontier and maintain solution diversity. Overall, the algorithms failed to identify the true Pareto frontier but this was dictated by the termination criteria set at 100 generations regardless of number of function calls used. An increase in the number of generations before convergence may drastically alter the ability of the algorithms to identify the true Pareto frontier.

When considered in the context of solution diversity, the No Classifier option performed poorly for all 6 Test Problems. Since the values are normalized for each Test Problem, a comparison cannot be made across Test Problems, but it can be seen from the presented results for M_2 that the No Classifier option is always out performed by one or more classifier options.

There is no single classifier that produces the greatest diversity measurement, M_2, across all the Test Problems. Instead, each classifier produces the most diverse population for at least one Test Problem. Random Forest and Naïve Bayes each produce the highest diversity for two Test Problems.

The tradeoff between the No Classifier option in terms of proximity to the true Pareto frontier and the classifier options in terms of solution diversity makes the choice of a "best" option difficult. The designer must choose between solution diversity and proximity. The inclusion of function count may aid in the choice of proximity or diversity. Perhaps the best choice is neither and instead a classifier that performs "okay" for both metrics and has a low function count.

The analysis of different machine learning classifiers for different problem types yielded high values for precision and recall. As previously mentioned, the classifier is retrained each generation using all previously evaluated designs as training data. The abundance of training data as the algorithm reaches convergence likely aids in the ability of the classifiers to accurately determine "good" and "bad" designs.

A result not specifically covered in this work, is the continued decrease in objective function evaluations when a classifier is included. In general, the number of objective function calculations was reduced by a considerable amount. While in these Test Problems the incorporation of a classifier is superfluous and increases computation time due to repeated classifier training, modern complex engineering problems require non-trivial functions. In these instances, the reduction in objective function calculations is necessary for the timely identification of non-dominated designs.

A requirement of this analysis is knowledge of the Pareto frontier shape *a priori*. While this allowed for the determination of the most appropriate machine learning classifier of the tested classifiers, it does not provide a means for predicting the shape of a Pareto frontier. Rather, future work is needed to accurately predict the shape of a Pareto frontier and identify the exact problem characteristics that result in such a shape. This will aid in the selection of a machine learning classifier.

REFERENCES

[1] M. Hauschild and M. Pelikan, "An Introduction and Survey of Estimation of Diistribution Algorithms," *Swarm and Evolutionary Computation,* vol. 1, no. 2011, pp. 111-128, 2011.

[2] P. Backlund, D. Shahan and C. 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, 2015.

[3] K. Zeliff, W. Bennette and S. Ferguson, "Multi-Objective Composite Panel Optimization Using Machine Learning Classifiers and Genetic Algorithms," in *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference,* Charlotte, 2016.

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

[7] 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.

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

[9] P. Larranaga, "A Review on Estimation of Distribution Algorithms," in *Genetic Algorithms and Evolutionary*

Computation, New York, Springer Science+Business Media, 2002, pp. 57-100.

[10] 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.

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

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

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

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

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

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

[17] E. Zitzler, K. Deb and L. Thiele, "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evoluationary Computation,* vol. 8, no. 2, pp. 173-195, 2000.

[18] S. Baskar and P. N. Suganthan, "A Novel Concurrent Particle Swarm Optimization," in *Proceedings of the 2004 Congress on Evolutionary Computation*, Portland, 2004.

[19] J. Vesterstorm and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Proceedings of the 2004 Congress on Evolutionary Computation*, Portland, 2004.

[20] E. Zitzler and L. Thiele, "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach," *IEEE Transactions on Evolutionary Computation,* vol. 3, no. 4, pp. 257-271, 1999.

[21] E. Zitzler and L. Thiele, "Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study," in *Fifth International Converence on Parallel Problem Solving from Nature*, Berlin, 1998.

[22] M. K. Buckland and G. Fredric, "The Relationship Between Recall and Precision," *JASIS,* vol. 45, no. 1, pp. 12-19, 1994.