ENME 625
Multi-Objective Optimization with Genetic Algorithms

Ben Knisely, Stephanie Martin, Jeff Twigg

May 9, 2017

Contents

1 Introduction

This report develops multi-objective optimization methods using genetic algorithms. Three cases of multi-objective optimization are considered: unconstrained optimization, constrained optimization, and robust optimization. A fitness function for the first two cases was developed using the built-in MATLAB 'ga' routine. The last case was developed such that uncertainty in function parameters could be handled and robustly optimized against.

Each method developed was evaluated through the use of well-known test problems. The resulting Pareto frontier of each method was evaulated in terms of quality metrics. Finally, the results of the developed methods were compared to results obtained by using the built-in MATLAB 'gamultiobj' routine.

2 Unconstrained MOGA Problems

The method by which fitness was assigned to each individual within a population was the well-known Non-dominated Sorting Genetic Algorithm (NSGA), which is fully described in [1]. For brevity, the full algorithm will not be described here, however the method shall be discussed.

NSGA evaluates the fitness of a population in the follow manner: first the current population is sorted into non-dominated layers. This may be accomplished in many ways, however the method used for this project was to continuously update the comparison population. That is, once a non-dominated layer is found, these population points are removed from the sorting algorithm and the next non-dominated layer is identified.

Once each individual belongs to a layer, the fitness assignment is accomplished layer-by-layer. The initial fitness of the first layer is assigned based on the number of points in the entire population. The similarity of an individual to other individuals in the current layer is evaluated based on the maximum and minimum values within that layer. This may be done in terms of the objective or the design space. For this project, similarity was computed in the objective space. Finally, based on the similarity of each individuals, a niche count is assigned to that individual and the final shared fitness of that individual is assigned by the ratio of the initial layer fitness and the individual niche count. The next layer is then evaluated in the same manner with its initial fitness based on the minimum shared fitness value of the previous layer.

Within NSGA, there are certain parameters which affect the final generation of the Pareto frontier. It was found for the test problems, the selection of the fitness degradation parameter, ϵ, and the sharing parameter, σ, were the most important. The fitness degradation parameter, ϵ, affects the difference in initial fitness values between layers. The larger the value of ϵ, the higher a penalty is

placed on individuals in non-dominated layers that are not in the Pareto layer. The sharing parameter, σ, affects the closeness of the final Pareto points. The higher the value of σ, the more "gaps" in the frontier. For each problem, these parameters were randomly tuned such that the "best" Pareto frontier (in terms of coverage difference and Pareto spread).

The next sections shall discuss the unconstrained problems ZDT1, ZDT2, and ZDT3, the results from the fitness function NSGA and MATLAB's "ga" function as well as the results from MATLAB's "gamultiobj" function for each.

2.1 ZDT1

The first test problem, denoted as ZDT1 in [1] is shown below.

$$
\begin{aligned}
\text{Minimize} \quad & f_1(\mathbf{x}) = x_1 \\
\text{Minimize} \quad & f_2(\mathbf{x}) = g(x) * h(x) \\
\text{where} \quad & g(x) = 1 + \frac{9}{(n-1)} \sum_{i=2}^{n} x_i \\
& h(x) = 1 - \sqrt{\frac{f_1(x)}{g(x)}} \\
& n = 30 \\
& 0 \leq \mathbf{x} \leq 1
\end{aligned}
$$

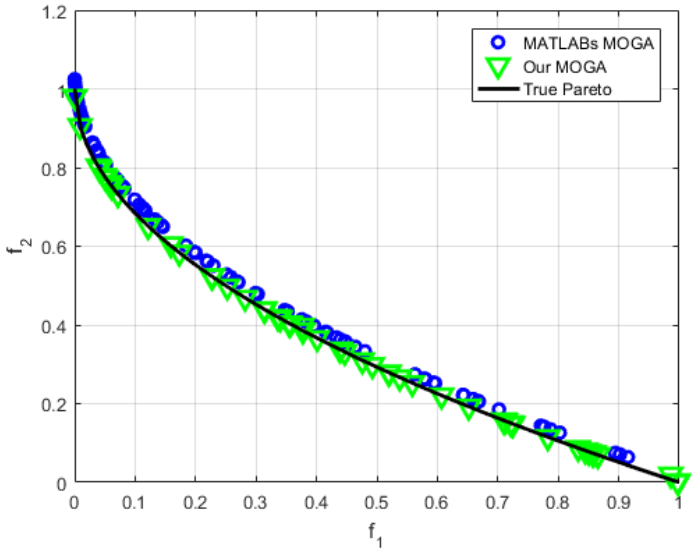
The true Pareto frontier for this problem occurs when $x_i = 0$ for i = 2,...,30. Figure 1 shows a sample result from both MATLABs built in MOGA and the MOGA developed in this project. In this problem, $\alpha = 1$, $\sigma = 0.158$, and $\epsilon = 0.22$. The final results were found using 300 chromosomes in each population. The quality metrics chosen to evaluate this problem (and all other problems) are Coverage Difference (CD) and Pareto Spread (OS). Ten runs for each algorithm were performed using the same initialization parameters and the mean and standard deviation of each metric are tabulated in Table 1.

Table 1: Quality Metrics for ZDT1

Metric	MATLABs MOGA	Our MOGA
CD	0.3874 (0.0164)	0.3582 (0.0040)
OS	0.9605 (0.1088)	0.9283 (0.0928)

From these metrics, there is certainly a trade-off between MATLABs MOGA and the MOGA developed in this project. The coverage difference of the new MOGA is lower (and therefore superior) in this problem whereas the Pareto spread is improved when using MATLABs MOGA.

Figure 1: Example Pareto Results for ZDT1



2.2 ZDT2

The second test problem, denoted as ZDT2 in [1] is shown below.

$$
\begin{aligned}
\text{Minimize} \quad & f_1(\mathbf{x}) = x_1 \\
\text{Minimize} \quad & f_2(\mathbf{x}) = g(x) * h(x) \\
\text{where} \quad & g(x) = 1 + \frac{9}{(n-1)} \sum_{i=2}^{n} x_i \\
& h(x) = 1 - \frac{f_1(x)}{g(x)}^2 \\
& n = 30 \\
& 0 \leq \mathbf{x} \leq 1
\end{aligned}
$$

The true Pareto frontier for this problem, similar to ZDT1, occurs when $x_i = 0$ for i = 2,...,30. Selected parameters were similar to ZDT1 where $\alpha = 0.5$, $\sigma = 0.5$, $\epsilon = 0.22$; the number of chromosomes was 300. Figure 3 shows a sample result from both MATLABs built in MOGA and the MOGA developed in this project. Table 2 summarizes the mean quality metrics for each algorithm over ten runs.

Figure 2: ZDT1 Quality Metrics

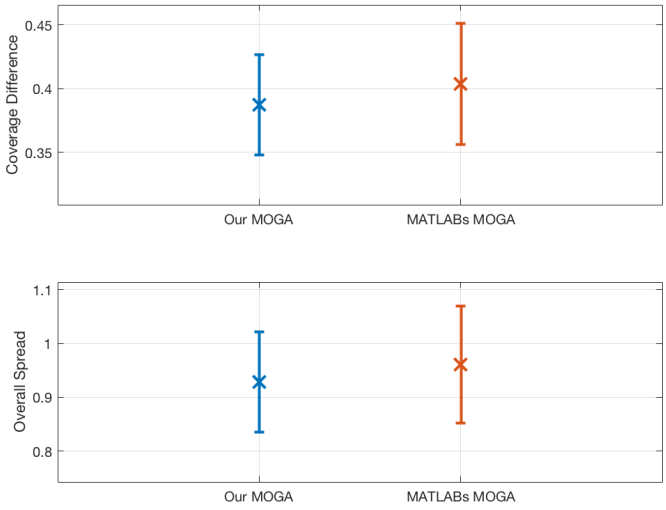


Figure 3: Example Pareto Results for ZDT2

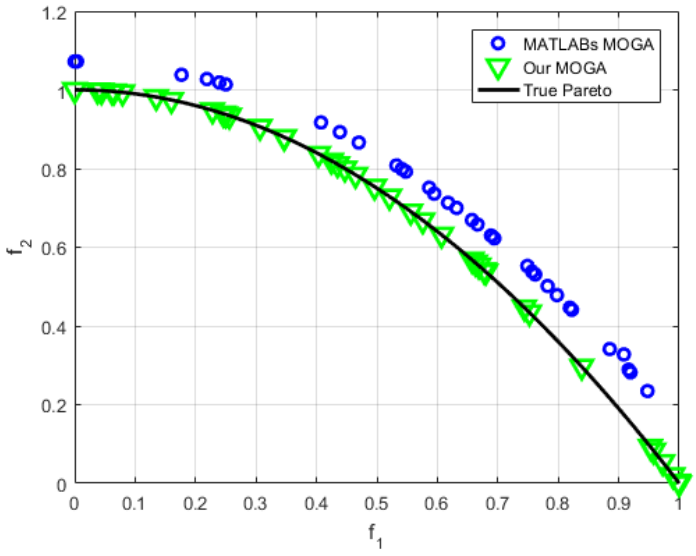
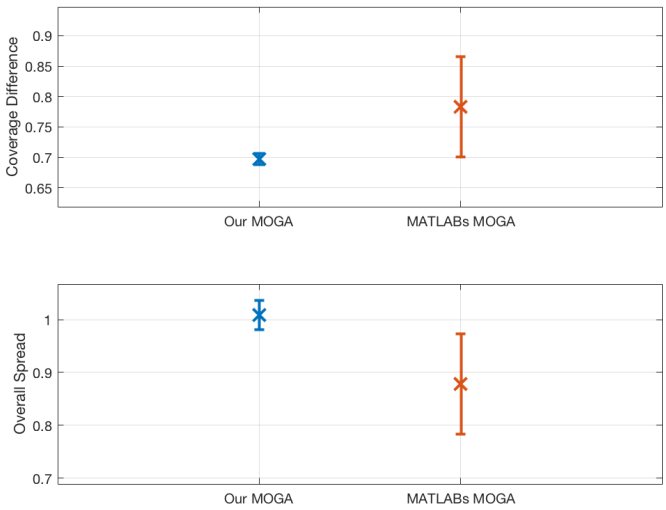


Table 2: Quality Metrics for ZDT2

Metric	MATLABs MOGA	Our MOGA
CD	0.7832 (0.0821)	0.6971 (0.0094)
OS	0.8781 (0.0946)	1.0086 (0.0278)

Figure 4: ZDT2 Quality Metrics



2.3 ZDT3

The third test problem, denoted as ZDT3 in [1] is shown below.

$$
\begin{aligned}
\text{Minimize} \quad & f_1(\mathbf{x}) = x_1 \\
\text{Minimize} \quad & f_2(\mathbf{x}) = g(x) * h(x) \\
\text{where} \quad & g(x) = 1 + \frac{9}{(n-1)} \sum_{i=2}^{n} x_i \\
& h(x) = 1 - \sqrt{\frac{f_1(x)}{g(x)}} - \frac{f_1(x)}{g(x)} \sin(10\pi f_1) \\
& n = 30 \\
& 0 \leq \mathbf{x} \leq 1
\end{aligned}
$$

The true Pareto frontier for this problem again occurs when $x_i = 0$ for i = 2,...,30. The fitness parameters selected in this problem were $\alpha = 1$, $\sigma = 0158$ and $\epsilon = 0.22$. Again, the number of chromosomes in this problem was 300. Figure 5 shows a sample result from both MATLABs built in MOGA and the MOGA developed in this project. Table 3 summarizes the mean quality metrics for each algorithm for ten runs.

Figure 5: Example Pareto Results for ZDT3

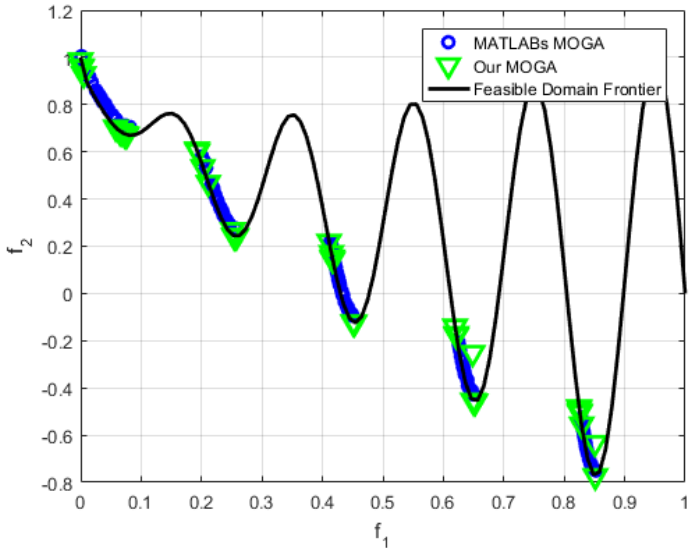
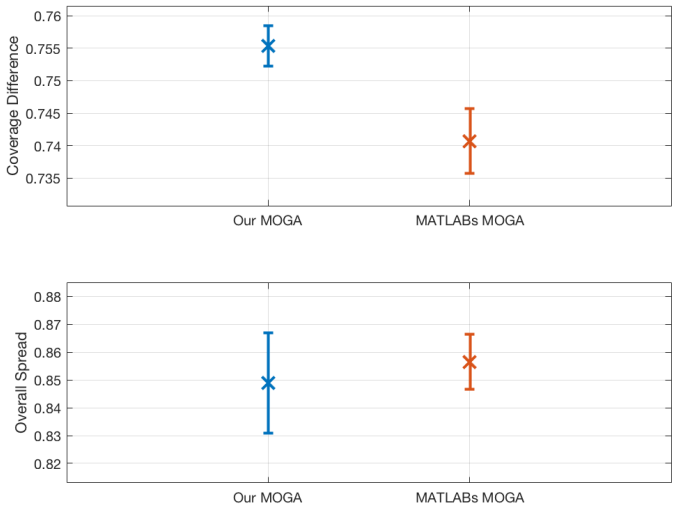


Table 3: Quality Metrics for ZDT3

Metric	MATLABs MOGA	Our MOGA
CD	0.7407 (0.0050)	0.7554 (0.0031)
OS	0.8565 (0.0098)	0.8489 (0.0180)

Figure 6: ZDT3 Quality Metrics



Overall, MATLABs MOGA outperforms the MOGA developed in this project in both coverage difference and Pareto spread. A paired t-test shows that the difference in the means for coverage difference is statistically significant (p¡0.05), while the difference is not statistically significant in Pareto Spread (p=0.21).

3 Constrained MOGA Problems

There are many ways in which fitness evaluation can be accomplished for constrainted problems. In general, some factors which may be included in fitness evaluation are the feasibility of an individual, the amount of infeasibility, and the number of violated constraints for a certain individual. Several methods to take into account these factors are described in [2]. For this project, all three parameters are used to evaluated the fitness of individuals in constrained problems. This method is denoted at CH-I4 in [2].

As in the unconstrained algorithm, there are fitness parameters which can be chosen to place more or less emphasis on any of the factors (i.e., amount of feasibility, etc.). For each test problem, these factors were chosen such that the "best" resulting Pareto frontier could be obtained. "Best", here is meant to describe the Pareto spread and coverage difference of the resulting frontier. The two factors which could be tuned in this algorithm were denoted as CF1 and CF2. CF1 is the factor affects the emphasis on the amount of infeasibility of an individual. Whereas CF2 affects the emphasis on the number of violated constraints.

The following sections shall discuss the test problems OSY, TNK, and CTP, the results from the constrained fitness function with MATLAB's "ga" function and from MATLAB's "gamultiobj".

3.1 OSY

This test problem, denoted as OSY in [1] is shown below, Figure 7 shows the true Pareto frontier for this problem. The factors chosen for this problem were $CF1 = 0.005$ and $CF2 = 0.015$, this placed a higher weight of the number of constraint violations than the amount of infeasibility.

$$
\begin{aligned}
\text{Minimize} \quad & f_1(\mathbf{x}) = -(25(x_1-2)^2 + (x_2-2)^2 + (x_3-1)^2 + (x_4-4)^2 + (x_5-1)^2) \\
\text{Minimize} \quad & f_2(\mathbf{x}) = \sum_{i=1}^{6} x_i^2 \\
\text{Subject to} \quad & g_1(x) = 1 - \frac{x_1 + x_2}{2} \leq 0 \\
& g_2(x) = \frac{x_1 + x_2}{6} - 1 \leq 0 \\
& g_3(x) = \frac{x_2 - x_1}{2} - 1 \leq 0 \\
& g_4(x) = \frac{x_1 - 3x_2}{2} - 1 \leq 0 \\
& g_5(x) = \frac{(x_3 - 3)^2 + x_4}{4} - 1 \leq 0 \\
& g_6(x) = 1 - \frac{(x_5 - 3)^2 + x_6}{4} \leq 0 \\
& 0 \leq x_1, x_2, x_6 \leq 10 \\
& 1 \leq x_3, x_5 \leq 5 \\
& 0 \leq x_4 \leq 6
\end{aligned}
$$

Figure 7: True Pareto Frontier for OSY

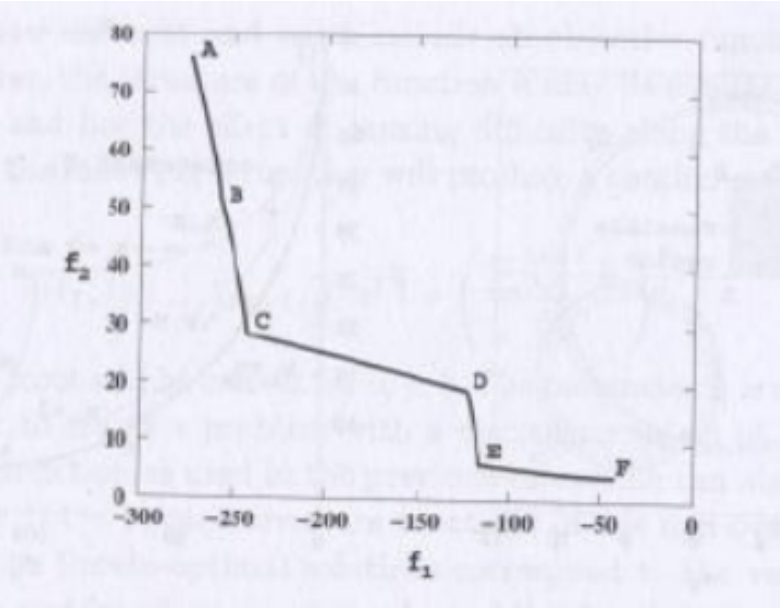


Figure 8 shows a sample result from both MATLABs built in MOGA and the MOGA developed in this project. Table 4 summarizes the mean quality metrics for each algorithm for ten runs.

Table 4: Quality Metrics for OSY

Metric	MATLABs MOGA	Our MOGA
CD	0.7882 (0.1995)	0.5507 (0.0685)
OS	0.5322 (0.3393)	0.9608 (0.2427)

Figure 8: Example Pareto Results for OSY

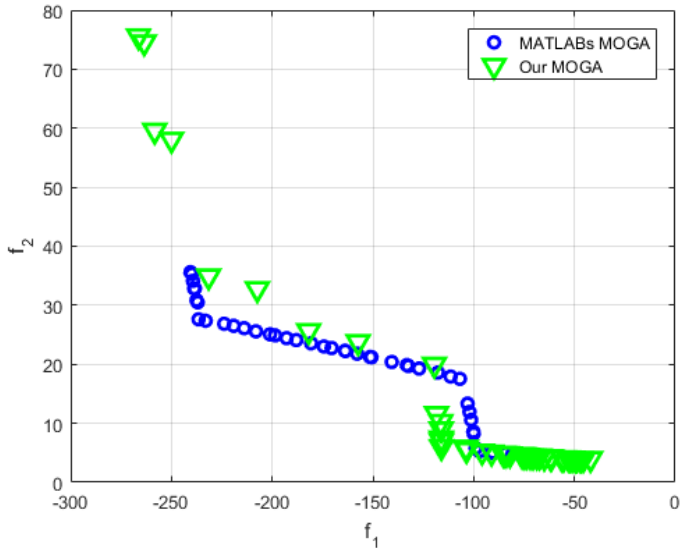
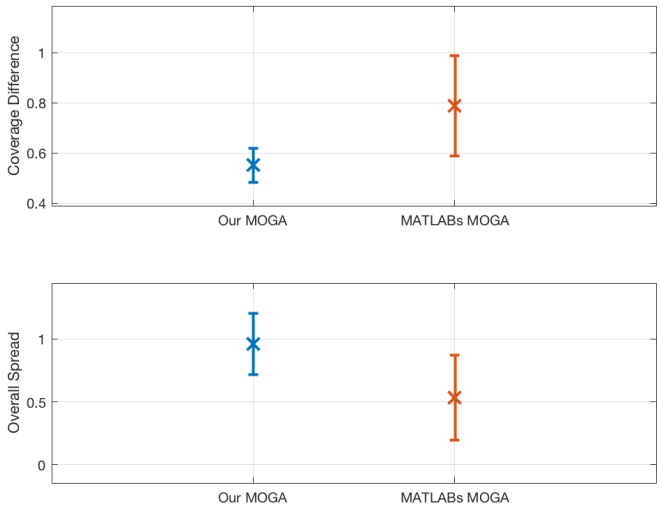


Figure 9: OSY Quality Metrics



3.2 TNK

This test problem, denoted as TNK in [1] is shown below with the true Pareto frontier shown in Figure 10. Constraint parameters CF1 and CF2 were both chosen to be 0.01 to place equal emphasis on amount of infeasibility and number of violated constraints.

$$
\begin{aligned}
\text{Minimize} \quad & f_1(\mathbf{x}) = x_1 \\
\text{Minimize} \quad & f_2(\mathbf{x}) = x_2 \\
\text{Subject to} \quad & g_1(x) = -x_1^2 - x_2^2 + 1 + 0.1\cos(16\arctan(\frac{x_1}{x_2})) \leq 0 \\
& g_2(x) = (x_1 - 0.5)^2 + (x_2 - 0.5)^2 - 0.5 \leq 0 \\
& 0 \leq x_1, x_2 \leq \pi
\end{aligned}
$$

Figure 10: True Pareto Frontier for TNK

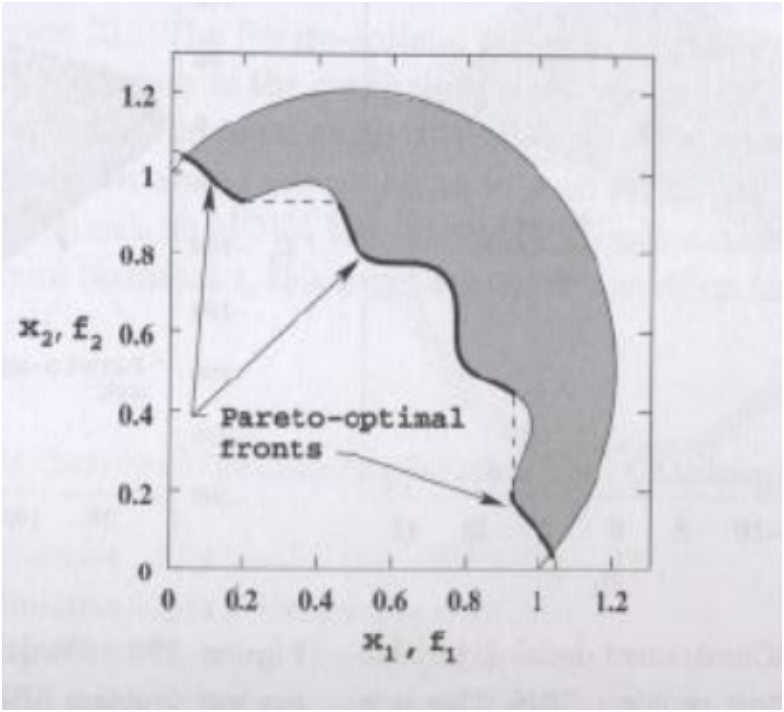


Figure 11 shows a sample result from both MATLABs built in MOGA and the MOGA developed in this project. Table 5 summarizes the mean quality metrics for each algorithm for ten runs.

Figure 11: Example Pareto Results for TNK

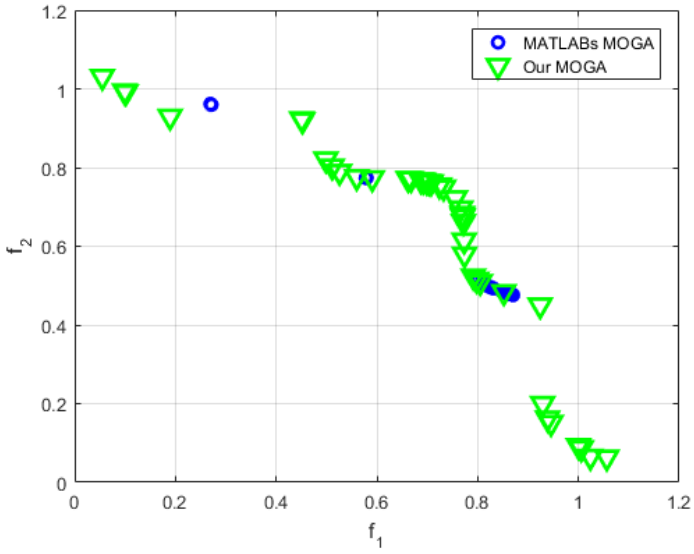
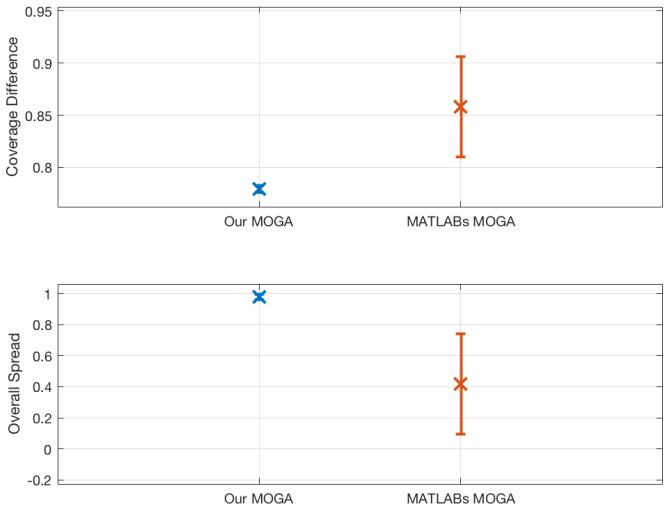


Table 5: Quality Metrics for TNK

Metric	MATLABs MOGA	Our MOGA
CD	0.8581 (0.0480)	0.7792 (0.0036)
OS	0.4177 (0.3216)	0.9763 (0.0178)

Figure 12: TNK Quality Metrics



Comparing the estimated Pareto frontiers to the true Pareto frontier, is is clear that our MOGA outperforms MATLAB's MOGA. The Pareto spread in our MOGA is significantly higher however the coverage difference is higher in MATLABs MOGA (p¡0.05).

3.3 CTP

This test problem, denoted as CTP in [1] is shown below and the true Pareto frontier is shown in Figure 13. Just as in TNK, constraint parameters CF1 and CF2 were both chosen to be 0.01 to place equal emphasis on amount of infeasibility and number of violated constraints.

$$
\begin{aligned}
\text{Minimize} \quad & f_1(\mathbf{x}) = x_1 \\
\text{Minimize} \quad & f_2(\mathbf{x}) = g(x)(1 - \sqrt{\frac{f_1(x)}{g(x)}} \\
\text{Subject to} \quad & g_1(x) = a|\sin(b\pi(\sin(\theta)(f_2(x) - e) + \cos(\theta)f_1(x))^c)|^d \\
& \quad - \cos(\theta)(f_2(x) - e) - \sin(\theta)f_1(x) \leq 0 \\
\text{where} \quad & \theta = -0.2\pi, a = 0.2, b = 10, c = 1, d = 6, e = 1 \\
& g(x) = |1 + (\sum_{i=2}^{10} x_i)^{0.25}| \\
& 0 \leq x_1 \leq 1 \\
& -5 \leq x_i \leq 5, i = 2, ..., 10
\end{aligned}
$$

Figure 13: True Pareto Frontier for CTP

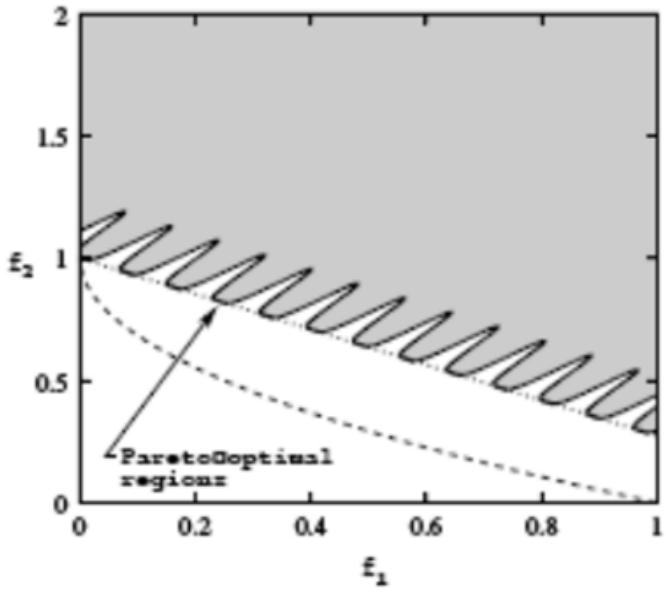


Figure 14 shows a sample result from both MATLABs built in MOGA and the MOGA developed in this project. Table 6 summarizes the mean quality metrics for each algorithm for ten runs [1].

Table 6: Quality Metrics for CTP

Metric	MATLABs MOGA	Our MOGA
CD	0.6802 (0.0067)	0.6802 (0.0092)
OS	0.7901 (0.0734)	0.5959 (0.1324)

Figure 14: Example Pareto Results for CTP

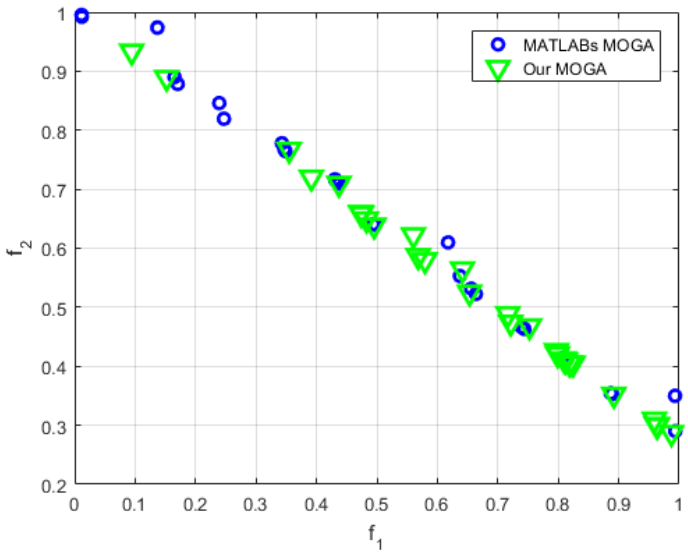
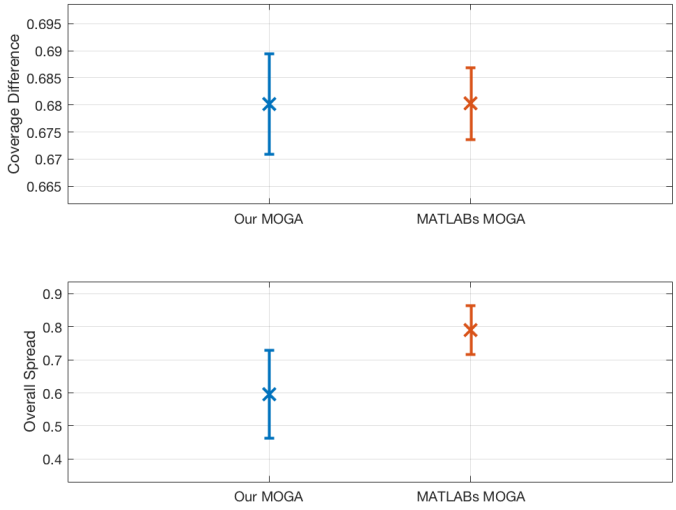


Figure 15: CTP Quality Metrics

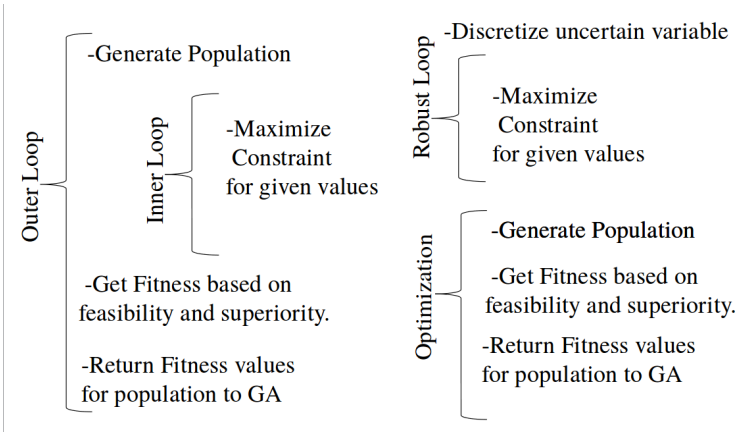


The mean values for coverage difference are exactly the same for both algorithms. In fact a paired t-test also shows the means are not statistically difference(p¿0.05). For Pareto spread, MATLAB's MOGA performs the highest.

4 Robust Problems

To solve problems with an uncertain variable we implement RMOGA. One way to ensure robustness for a given population considered in GA, is by finding the value of the uncertain variable which maximizes the constraint values acording to each member in the population. Another way is to discretize the dependent variable and then find which uncertain variable in this set maximizes the constraint values. This latter method is the one we chose because of the decreased execution time over the former. These two options are shown in figure 16. Since the uncertainty term only appears in the constraints portion of both of the problems we consider this discretization is fairly simple. In addition, since the uncertain terms are linearly combined in the constraints, this discretization is trivial.

Figure 16: Options for making optimization robust to uncertain variables.



4.1 Robust TNK

This test problem, denoted as TNK in [1] is shown below with the true Pareto frontier shown in Figure 10.

$$
\begin{aligned}
\text{Minimize} \quad & f_1(\mathbf{x}) = x_1 \\
\text{Minimize} \quad & f_2(\mathbf{x}) = x_2 \\
\text{Subject to} \quad & g_1(x) = 1 + 0.1\cos(16\arctan(\frac{x_1}{x_2})) + \\
& \qquad 0.2\sin p_1 \cos p_2 - x_1^2 - x_2^2 \leq 0 \\
& g_2(x) = (x_1 - 0.5)^2 + (x_2 - 0.5)^2 - 0.5 \leq 0 \\
& 0 \leq x_1, x_2 \leq \pi \\
& -1 \leq p \leq 3
\end{aligned}
$$

Figure **??** shows a sample result from both MATLABs built in MOGA and the robust MOGA developed in this project. Table 8 summarizes the mean quality metrics for each algorithm for ten runs.

Figure 17: Example Pareto Results for TNK

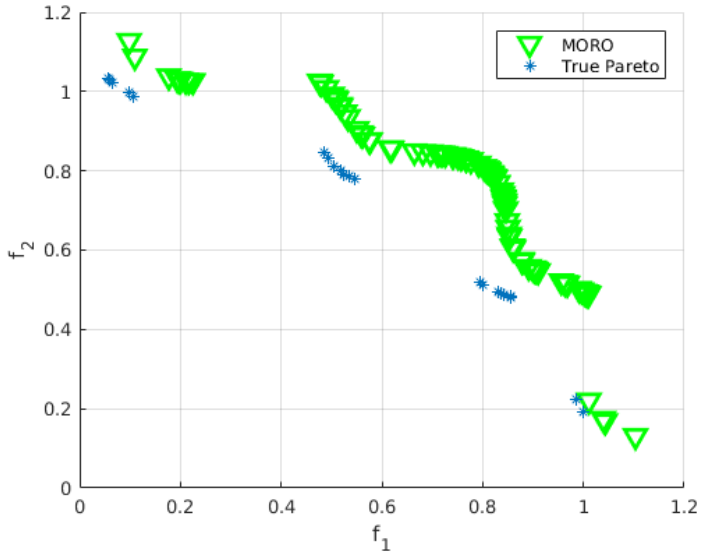


Table 7: Quality Metrics for Robust TNK (10 Runs)

Metric	Our MOGA
CD	0.95 (0.014)
OS	0.66 (0.012)

Comparing the estimated Pareto frontiers to the true Pareto frontier, is is clear that our MOGA outperforms MATLAB's MOGA. The Pareto spread in our MOGA is significantly higher however the coverage difference is higher in MATLABs MOGA (p¡0.05).

4.2 Flight Planning Problem (FPP)

Now let us consider the final problem where we try to minimize the total flight time from the start location $(0, 12.5)$ to the finish location $(40, 12.5)$. In this problem there is a know wind velocity field which changes the total velocity of the aircraft with respect to the ground. At the same time, we want to maximize the distance of the flight path from some group of some exclusion zones. This also introduces a constraint that requires the flight path not intersect the exclusion zone. Each exclusion zone is approximated by a circle with known centers. The radius of each exclusion zone is known to $\pm 2m$. As a result this problem can be approached as a bi-objective optimization with a robust feasibility constraint.

$$
\begin{aligned}
\text{Minimize} \quad & f_1(\mathbf{x}, \mathbf{y}) = flightTime(\mathbf{x}, \mathbf{y}) \\
\text{Maximize} \quad & f_2(\mathbf{x}, \mathbf{y}) = -\min_{i,j} \sqrt{(\mathbf{x}_i - \mathbf{xc}_j)^2 + (\mathbf{y}_i - \mathbf{yc}_j)^2} \\
\text{Subject to} \quad & \mathbf{g}_j(\mathbf{x}, \mathbf{y}) = -\min_{j} \left[\mathbf{r}_j + p_0 + \Delta p - \sqrt{(\mathbf{x}_i - \mathbf{xc}_j)^2 + (\mathbf{y}_i - \mathbf{yc}_j)^2} \right] \leq 0 \\
\text{where} \quad & 0 \leq x \leq 50, i = 2, ..., 10 \\
& 0 \leq y \leq 25, i = 2, ..., 10 \\
& p_0 = 0 \\
& \Delta p \in [0, 2]
\end{aligned}
$$

The function $flightTime(\mathbf{x}, \mathbf{y})$ is a black-box. The values $\mathbf{xc}_j$ and $\mathbf{yc}_j$ specify the exclusion zone centers while $\mathbf{r}_j$ specify the radii. A solution to this problem is shown in figure 18. In this figure there are two exclusion zones.

Figure 18: True Pareto Frontier for CTP

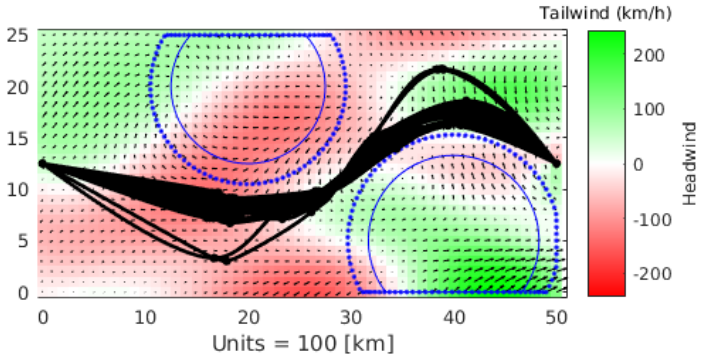


Figure 19: True Pareto Frontier for CTP

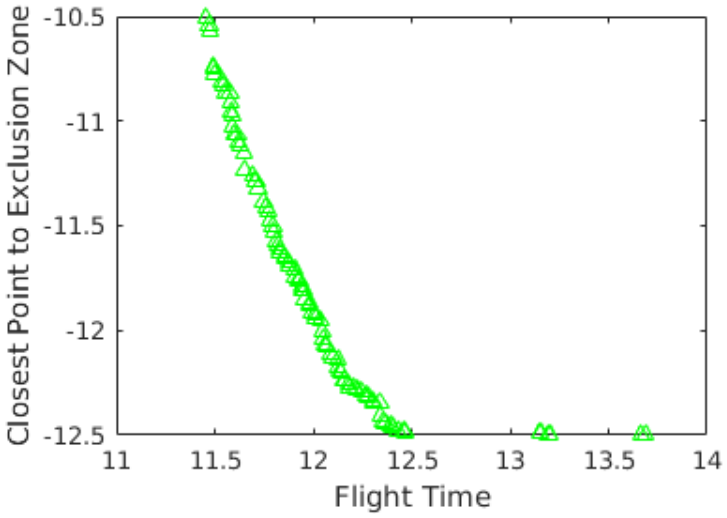


Table 8: Quality Metrics for Robust FPP (10 Runs)

Metric	Our MOGA
CD	0.2427 (0.0704)
OS	0.4794 (0.1220)

5 Conclusion

Overall, it was found that different optimization algorithms result in a range of solutions for a given test problem. Thus it is important, when developing an optimization algorithm, to test against a range of optimization problems. In addition, it was also concluded that special attention should be paid to tunable parameters within each algorithm. Understanding the nature of the parameter and how it affects the algorithm is crucial to obtaining the "true" Pareto frontier.

References

[1] K. Deb, "Multi-objective optimization using evolutionary algorithms, 2001," *Chicheter, John-Wiley.*, 2001.

[2] A. Kurpati, S. Azarm, and J. Wu, "Constraint handling improvements for multiobjective genetic algorithms," *Structural and Multidisciplinary Optimization*, vol. 23, no. 3, pp. 204–213, 2002.