An Analysis of NeuroEvolution Through Augmenting Topologies in the Optimization of Aircraft's Airfoils

Thousand Oaks High School

April 2019

Word Count: 4971

Table of Contents

Abstract

The use of Neuroevolution of Augmenting Topologies (NEAT) algorithm was tested for its effectiveness at optimizing the airfoil lift-drag coefficient at varying degrees of control over the design and was then compared to a traditional Multidisciplinary Design Analysis and Optimization (MDAO) algorithm. The experimentation was performed with three different degrees of control: single, six, and 211. Each test was run 30 times to ensure a statistically significant sample size and then was analyzed with an ANOVA and Kruskal-Wallis test to assess for statistical differences. In the results of the study, NEAT with six variables of control optimized orders of magnitude more than any of the other optimization experiments. Furthermore, the study reveals that NEAT yielded a greater improvement than MDAO with the same degree of control; however, NEAT had a longer runtime and inconsistent results.

Introduction

Before the advent of computers capable of modeling the complexities of fluid dynamics for three-dimensional objects, aircraft optimization occurred by creating a physical design and testing the design's flight properties in a wind tunnel (Stack, 1935). This process was highly cost prohibitive due to material costs and lengthy construction time. As result, limiting the amount of testing performed and making rapid improvements in flight performance less likely to occur (Gellispie, 2004). With continued advances in computational power and efficiency, the optimization of aircraft now relies more on computer-based methods of optimization (Sobieszczanski-Sobieski & Haftka, 1996). A major drawback of these types of methods was

improving model accuracy required a subsequent increase in the complexity of the differential equations used which elevated the computational cost (Gellispie, 2004). The main limiting factor in these methods was the immense amount of computing power required to provide an accurate and High Fidelity model. Yet, in 1998 a computer-based method was still a third the price of the previous physical-based method (Deloachl, 1998). A commonly used method of optimization, Multiple Disciplinary Analysis and Optimization (MDAO), uses an iterative process along with a minimize function to find ideal values for the airplane design variables (Gray, Moore, & Naylor, 2010). Though MDAO has been an effective method of optimization, it requires a substantial amount of computing power and it does not always find the optimal solution (Secco & Mattos, 2015).

New Computer Algorithms

Over the last 20 years, the trend, known as Moore's Law, in which the number of transistors and correlatively computing power in a microcomputer chip doubled every two years, allowed computational methods to become the primary method of optimization (Moore, 1998). This advancement in computing power allowed previously theoretical Artificial Intelligence algorithms such as Artificial Neural Networks (ANNs) to be applied for the first time (Pitts, 1942). ANNs replicate the function of the human brain by creating a network of interconnected nodes or neurons with both an input and output with three different types of layers: input, hidden, and output (Fig. 1). Furthermore, the hidden layer is an optional component in ANNs

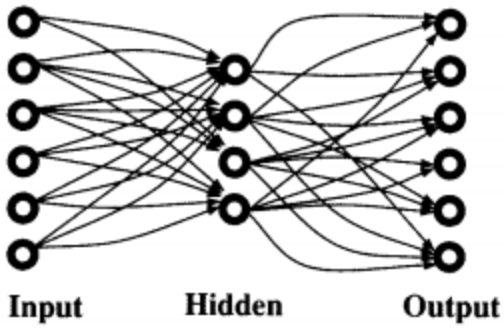

Figure 1. A diagram of a basic ANN (Dormehl, 2019)

with the user determining the size and number of hidden layers. The application of ANNs was an attempt to replicate human intelligence using a computer algorithm, and the idea behind ANNs was if the brain could be modeled mathematically, then a computer might be able to think and learn similar to a human allowing it to devise solutions previously only possible by humans (Bos, 1998). Though the goal of achieving intelligence similar to that of a human has not been achieved yet, ANNs are still an effective algorithm for modeling, predicting, and optimizing. Since ANNs create a mathematical model of a brain, biological functions are converted into a mathematical representation. The neurons in a brain form connections with each other and the more important a connection, the stronger it becomes. Similarly, the connection between nodes in an ANN is given a numerical value called weight which is representative of the importance of that input. Then inputs are multiplied by the weight adjusting its value (Basheer & Hajmeer, 2000). Also, in an ANN, each node is assigned some numerical value called bias which changes the likelihood of a node firing or returning a value. The output of a node is determined through the evaluation of an activation function at the summation of all of the adjusted inputs and bias. The previously mentioned activation function is used to determine a node's response to its inputs and can be any function with two variables. The sum inside a node is equal to an X value on an activation function; the corresponding Y value is the output of the

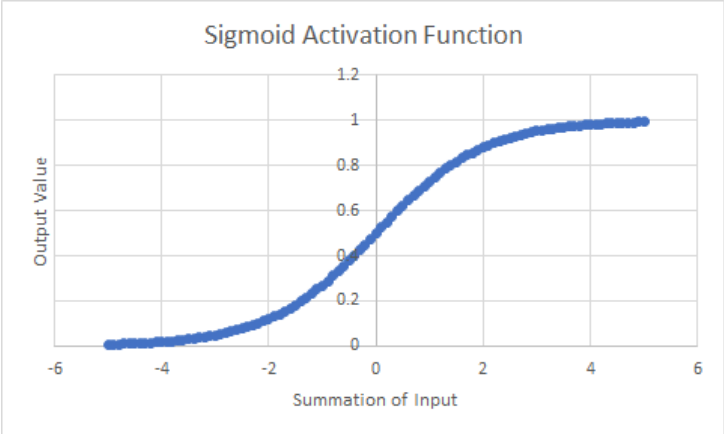

Figure 2. Example of an Activation Function, made in excel.

node (Fig. 2). After an ANN is given inputs, the nodes fire and then outputs are returned but, without the ANN being taught how to correlate the inputs with desired outputs, the algorithm is useless. In order to make ANNs useful, a process known as learning is used to teach an ANN to make the proper connections between inputs and outputs (Hochreiter, Younger, & Conwell, 2001).

ANN Learning Process for Modeling a Problem

The learning process for an ANN replicates how the human brain learns and understands problems. When a human is given a task and can evaluate performance, the brain adjusts the strength of connections between neurons based on which impacted performance the most. Whether it is learning algebra or how to spell, the connection between neurons will strengthen or weaken based upon whether the task was performed correctly. With continual repetition of a task, the brain keeps improving until it can consistently perform the task properly. Similarly, an ANN adjusts its weights and biases in order to model a problem accurately. In order to learn, it must be given a set of training data that is representative of the problem or scenario it is trying to model or solve. Using a training set, an ANN will calculate and then try to reduce a loss value. In most cases, it is equal to the absolute value of the ANN output subtracted by the expected output from training data (Hochreiter, Younger, & Conwell, 2001). As the loss value approaches zero, this reflects the more accurately the ANN is modeling the problem. Therefore, the goal of learning is to reduce the cost value as close to zero as possible through the modification of the weights and biases. Learning may occur using other methods.

Backpropagation Learning Method

Backpropagation is a commonly used learning method that works backward through an ANN to adjust the weights and biases after finding ideal weight and bias values. The ideal values are found through the creation of a loss function consisting of the weights and biases in an ANN as independent variables, while returning loss (Fig. 3) (Amari, 1993). Utilizing Gradient Descent, an optimization technique, the minimums of the cost function are found (Hochreiter, Younger, & Conwell, 2001). Gradient Descent works by starting at a point on the function and then calculating the slope or gradient in every direction (Amari, 1993). The initial values are then shifted a set interval in the direction that decreases the output, as indicated by the gradient values. The process is repeated until the gradients are equal to zero; meaning a minimum is found. Gradient descent is necessary as ANN's cost function can be increasingly complex, making it impossible to use traditional calculus to find minimums. However, a flaw with Gradient Descent is that a poor local minimum of the loss function could be found instead of the ideal values (Castillo et al., 2003).

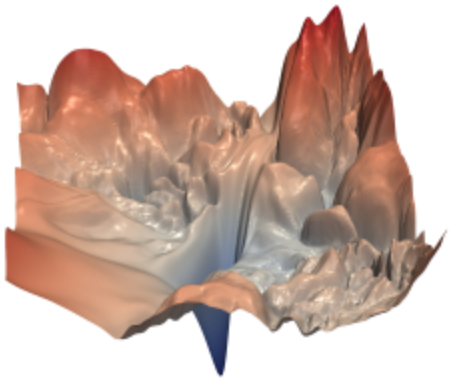

Figure 3. Visualization of the loss function (Castillo et al., 2003)

Evolutionary Algorithms to Prevent Stagnation at Poor Solutions

An evolutionary algorithm is any algorithm inspired by the Darwinian concept of evolution, as it replicates the process of natural selection (Veldhuizen & Lamont, 2000). A genetic algorithm, a frequently used evolutionary algorithm, performs optimization by creating

a population of randomly generated solutions and a genome of each member of the population, which represents values or aspects of that member analogous to DNA (Ray, Tai, & Seow, 2001). The format of the solutions and genome are dependent on the optimization being performed; it could be a route from different cities, an airplane design, or an answer to a problem. Then, each member of the population is tested for its fitness, or its ability to complete the given task and poorly performing members are removed (Veldhuizen & Lamont, 2000). The function responsible for calculating fitness is dependent on the problem being optimized. Subsequently, using the data from the best performing members' genome, new members are made by combining data from ideal members, using a function called crossover that replicates the process of combining of DNA in reproduction (Ray, Tai, & Seow, 2001). Furthering the similarities to Darwinian evolution, there is a chance of random changes occurring to a member referred to as a mutation (Shopova & Vaklieva-Bancheva, 2006). The process of evaluation, selection, and reproduction are repeated for some number of generations, and the fittest member are the ideal solution produced. In terms of airfoil modeling, the most ideal form are selected and reproduced.

Combining ANN and Evolutionary Algorithms in NeuroEvolution

NeuroEvolution is a type of evolutionary algorithm that utilizes a genetic algorithm, but instead of applying the evolutionary process to the solution, it controls weights and biases along with other parameters in the ANN, such as hidden layer size (Floreano et al., 2008). NeuroEvolution eliminates the need for Gradient Descent by using the evolutionary process to eliminate poorly performing values. Furthermore, it avoids the pitfalls of Gradient Descent, as

the chance of mutation prevents the weight and bias values from getting stuck at a poorly performing minimum (Secco & Mattos, 2015).

Literature Review

This study analyzes computer algorithms at optimizing the performance of airplane airfoil. The success of algorithms applied in other fields indicates potential when applied to aircraft.

Applications of ANN and Evolutionary Algorithms in Non-Aerospace Fields

Generative design (GD) is a design process that applies ANNs and or Evolutionary algorithms to generate more efficient designs that perform a task within certain constraints (Krish, 2011). GD algorithms are given a design such as a chair or a car part; then it will generate alternative designs (Fernandes, 2013). A simulation evaluates the alternative designs assigning them a fitness-based upon the performance of the design allowing the algorithm to improve either through selection or adjustment of weights and biases (Fernandes, 2013). The process is repeated for an user-determined number of generations. Figure 4 is an image of what a GD process applied to a chair could yield. In experimentation in which a GD process was applied to air conditioning fan designs to reduce energy use, the fan blades produced by GD had 58.5% improvement in energy efficiency

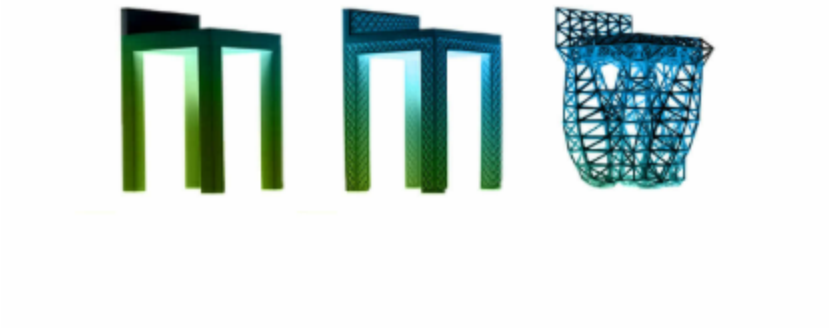

Figure 4. Example of generative design (GENERATIVE,2018)

(Berquist et al., 2017). Other research into the use of GD applied the algorithm to building designs and indicated great potential but only to simple designs or parts of buildings. A limitation with GD made optimization of complex building design currently impossible (Caldas, 2006). Complexity overwhelmed the algorithm making it too difficult to improve the overall design. In turn, intricate objects should be broken down into smaller parts making it easier to optimize (Caldas, 2006). Also, research into the application of GD on car body designs found that further research is needed to determine how to implement GD on car body parts due to the complexity of the problem (Gavacova & Veres, 2013). GD has been successfully applied in the optimization of simple problems with a high degree of success, but further research needs to be done into its application on complicated systems. However, the high degree of improvement with simple problems indicates possible success with similar types of algorithms.

ANNs and Evolutionary Algorithm in Aerospace Applications

ANN predicts how an aircraft's performance changes with different designs. The benefit of ANN is that it is far less computationally intensive than traditional methods of predicting performance. Instead of having to compute complex differential equations, ANN identifies trends with how inputs affect outputs which allows it to predict outputs based upon new inputs (Secco & Mattos, 2015). ANNs for predicting performance would use a database of airplane design and its performance for training data. These trends are realized by iterating through the training data adjusting weights based upon how specific values on the wing affect the performance of the design. Ideally, a large enough database enables an ANN to understand all the connections in design components allowing it to predict the performance of new designs effectively. Though the fidelity of predictions made by ANNs is not as good as that of other

simulations that use complex differential equations. If ANNs are effective at modeling the performance of aerospace designs, they may be effective at optimizing designs. Research has proven the usefulness of ANNs in aerospace fields. ANNs are effective in reducing the false alarm rate in onboard fault detection, isolation and estimation systems for high-performance aircraft which predicts how pilot's inputs affects performance in order to prevent structural failure (Chinag & Youssef, 1994). The analysis of a Multilayer Perceptron ANN, an ANN with a hidden layer, predicted the performance coefficients of transport aircraft. The study found a relatively simple network could effectively predict the coefficients of an airfoil and a complete airplane with a fixed geometry (Wallach et al., 2006).

Further research into the use of ANNs found that a special type of ANN, a Convolutional Neural network, is comparable to the wind tunnel and traditional computational fluid dynamics methods of performance prediction (Ignatyev & Khrabrov, 2018; Zhang, Sung, & Mavris, 2018). Also, research into applying evolutionary algorithms to optimize an aircraft's lift-drag coefficient (L/D) found that genetic algorithms were effective at finding the optimal design, but the time to converge on the design was not consistent (Holst, 2004). A comparison of an ANN and an evolutionary algorithm in optimizing an airfoil for optimal aerodynamic design determined that the ANN was faster at converging on the optimal solution. However, the study indicated more research is needed to determine if this holds true for all design cases (Rai, 2004). The effective use of ANN and evolutionary algorithms possibly indicates these types of algorithms are effective at modeling aerospace problems and further application of these algorithms in aerospace could outperform traditional methods.

Neuroevolution in Aerospace Optimization

Research into the application of Neuroevolution on aerospace design is limited and recent. In experimentation with an evolutionary algorithm which grows and optimizes the hidden layer of ANN to optimize airfoil aerodynamic efficiency, the algorithm reduced the computational time need by 60% (Timnak et al., 2017). Other research conducted used a special type of ANN called a Generative Adversarial network (GAN), which has ANNs competing with each other in a zero-sum game to optimize a task (Chen, Chiu, & Fuge, 2019). A GAN is not considered an evolutionary algorithm as it lacks the necessary components to replicate evolution, but the competitive aspects and use of ANN are similar to that of a NeuroEvolutionary algorithm. In the application of GANs to optimize the aerodynamic design of airfoils, GANs drastically reduced the time to converge on the optimal solution (Chen, Chiu, & Fuge, 2019). The improvement in efficiency observed with neuroevolutionary algorithms show their potential in aerodynamic optimization while the limited literature indicates a significant gap in aerospace research. For instance, the lack of testing with other types of neuroevolutionary algorithms such as Neuroevolution of Augmenting Topologies (NEAT) is major hole in current research.

Neuroevolution of Augmenting Topologies

NEAT is new type of neuroevolutionary algorithm which is similar to traditional ANNs in that it has an input, output and hidden nodes. However, instead of having uniform layers, the nodes are connected in non-uniform topological structures (Stanley, 2004). The genome of a NEAT algorithm consists of genes which represent a connection between two nodes. Each gene is given a specific identification label, which represents the same connection for all networks in

the population that have that gene. After evaluation and selection are performed, the crossover process begins in which the best networks reproduce by combining their genomes. When genes conflict with each other, such as when one network calls for a connection to be disabled and the other has it enabled, the gene from the fittest network is taken (Figure 5) (Stanley, 2004). During the reproduction process, there is a chance that a network will randomly mutate which creates or deletes a node or connection (Fig. 6). NEAT does not require backpropagation because during the crossover and mutation process the weights and biases are affected as well. Backpropagation is the most computationally expensive part of ANN; therefore, NEAT significantly reduces runtimes compared to traditional ANN (Stanley & Miikkulainen, 2002; Zhu et al., 2005). Experimentation with other traditional ANN and neuroevolutionary algorithms is

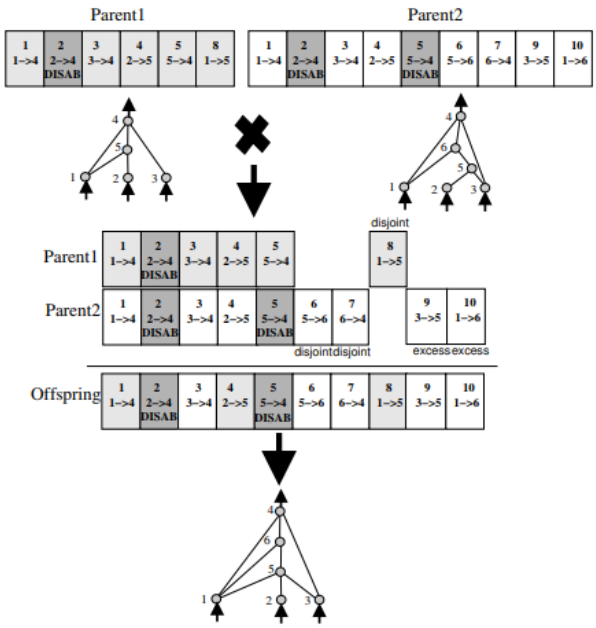

Figure 5. A visualization of NEAT crossover, the boxes represent the genome and the interconnected nodes represent the Network (Stanley, 2004)

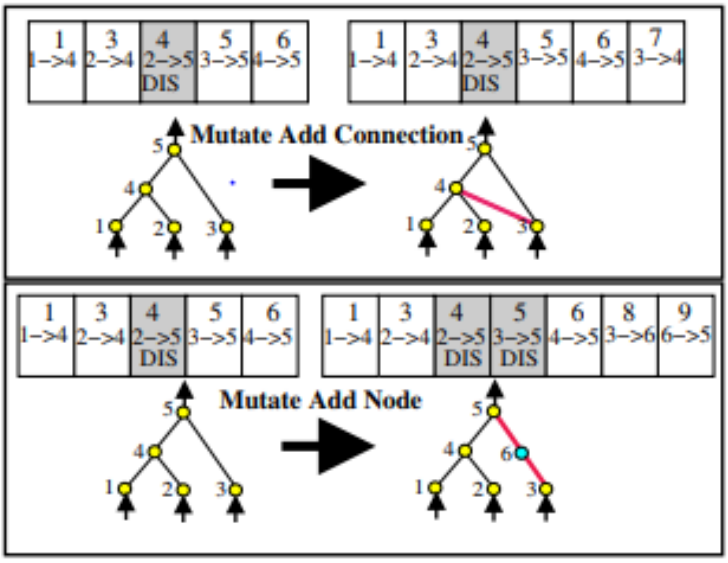

Figure 6. Visualization of NEAT mutation (Stanley, 2004)

effective in optimizing and modeling aerospace problems. However, due to the recent invention of NEAT networks, they have not been applied in aerodynamic optimization tasks yet.

Purpose

The study's purpose was to apply a NEAT algorithm to the optimization of an airplane's airfoil at varying degrees of control. By comparing the performance of a final design made by NEAT to the initial design, the study evaluated the improvements in efficiency created from a NEAT optimization. Also, the study compared the results of a NEAT method of optimization to results of a MDAO method of optimization. Furthermore, the study analyzed the results of NEAT algorithm on the final design in order to conclude whether it was an effective method at optimizing the airfoil of an airplane.

Research Question

How effective are NeuroEvolution of Augmenting Topologies algorithms at optimizing aerodynamic efficiency?

Hypothesis

Null: NeuroEvolution of Augmenting Topologies algorithms have no effect on the aerodynamic properties of airplane designs.

Alternative: NeuroEvolution of Augmenting Topologies algorithms are effective at optimizing the aerodynamic properties of airplane designs.

Methods

Program Design

The Python script in the study utilized the modules NEAT-python and OpenAeroStruct for the NEAT algorithms and aerospace analysis respectively (McIntyre et al., n.d.; Jasa, Hwang, & Martins, 2018). MDAO optimization was a built-in option in OpenAeroStruct, while the NEAT method of optimization was explicitly made for this study. For the NEAT networks in each experiment performed, the number of inputs and outputs corresponded with the number of variables controlled on an aircraft. In turn, the NEAT network for an experiment with six variables of control had six inputs and six outputs. This experiment design meant that for each network there was only one design and the network was evaluated on the performance of that design.

In contrast, previous ANN research in optimizing aerodynamic performance learned from an extensive library of designs and was trained to optimize any wing input. In order to evaluate the effectiveness of a design, the L/D was utilized. It is a ratio between the lift and drag generated by a wing at any speed or condition and is calculated by taking lift generated by a wing at a given time and dividing it by the drag produced by the wing at the same time (Smetana et al., 1975). L/D provides a single value that adequately represents the performance of an airplane as the higher the lift is, the slower an aircraft can fly while still generating sufficient lift. While the lower the drag is, the less force would be needed to allow an aircraft in order to reach a given speed. The fitness function, which calculates the effectiveness of a design, remained constant among all experiments and was computed by subtracting the initial design's L/D from the final design's L/D or Fitness = L/D_F - L/D_I (Smetana et al., 1975). Therefore, the NEAT network's

were optimizing the L/D of the designs. Furthermore, when a network's design performed poorly and failed to improve, it was removed from the population and replaced with a new network following the evolutionary process described previously.

Experiment Design

In the study, three different degrees of control were tested: one, six, and 211 variables. Both MDAO and NEAT tested one and six variables. Due to a limitation with the MDAO code, only the NEAT optimization was performed with 211 variables of control. All experiments had an initial L/D of 26.2.

In NEAT optimization when run over a finite number of generations, the resulting efficiency was inconsistent. Therefore, to perform a proper analysis, each NEAT algorithm was run 30 times to get a statistically significant sample size. The MDAO returned the same design every time the optimization was performed. The degree of optimization by MDAO was constant making it unnecessary to run the simulation more than once.

Table 1 defined what variables were optimized at each degrees freedom. The following variables were optimized: angle of attack, alpha maneuver, spar thickness, wing geometry, skin thickness, wing twist and mesh points. The angle of attack is the angle the wing makes relative to the ground, alpha maneuver is the range in which the angle of attack can change during flight, spar thickness refers to the thickness of the wing's spars, wing geometry is the wing thickness divided by chord thickness, wing skin thickness is the thickness of metal around the frame of the wing, and wing twist is representative of how much the wing tip curves up (Jasa, Hwang, & Martins, 2018). The 205 mesh points are all the points on the wing if it was plotted in three dimensionally.

Table 1. The variables on the wing controlled by the optimization method at each degree of control.

	One Design Variable	Six Design Variables	211 Design Variables
Angle of Attack	Used	Used	Used
Alpha Maneuver		Used	Used
Spar Thickness		Used	Used
Wing Geometry		Used	Used
Wing Skin thickness		Used	Used
Wing Twist		Used	Used
All 205 mesh points			Used

Parameters in Configuration File

In designing the experiment, all variables were held constant except for the degree of control and the type of optimization. Each NEAT optimization was performed with a population of 10 and over 30 generations. MDAO does not have a population or generational parameter, so the parameters for the evolutionary process only applies to NEAT experiments. The maximum

fitness was set to infinity. All NEAT optimization used an inverse activation function. In accordance with preliminary testing, the number of hidden nodes was 20 for a single variable, 150 for six variable and 400 for 211 variable. For all other NEAT parameters, see Appendix A.

Statistical Analysis

In order to effectively compare and establish a difference in results for the different methods of optimization and at varying degrees of control, the Kruskal-Wallis and Anova test were utilized in excel spreadsheets. If the Kruskal-Wallis and Anova test return p-values indicating a difference in significance, the kurtosis and skewness values for each data set were analyzed to determine which value to accept. Since Anova testing assumes the data is distributed normally, the Anova p-value would be accepted if the data is distributed normally. Otherwise, the non-parametric Kruskal-Wallis p-value will be accepted as it does not assume normal distribution.

Results

After running all 30 optimizations for each method, NEAT with six variables yielded the most significant improvement over the initial design and NEAT with 211 variables having the worst individual optimization (Table 2). The six variable NEAT had the most substantial average improvement followed by 211 variable NEAT and with the lowest average being single variable MDAO (Table 2). Due to the high variance in the NEAT 6 and 211 variable experiments, the averages are inflated by outliers, but the trimmed average takes the average with the outliers removed. The trimmed averaged decreased the value of all NEAT methods, six

variable NEAT fitness was decreased from 3777 to 127, but the ranking of algorithms based on performance did not change (Table 2). The average runtime for both types of algorithms increased with an increase in the number of variables, and the NEAT algorithm always took longer than the MDAO (Table 2).

Table 2. Different methods and degrees of control results

	NEAT One	NEAT Six	NEAT 211	MDAO One	MDAO Six
Max Fitness L/D	14.5	109,029	151.5	1.1	6.23
Min Fitness L/D	.4	-1	-18	1.1	6.23
Average Fitness L/D	4.76	3777.37	16.90	1.1	6.23
Variance	16.06	$3.95*10^8$	1481.90	0	0
Trimmed Average Fitness L/D	4.33	127.28	11.22	1.1	6.23
Percent that regressed	0%	3.3%	33.3%	0%	0%
Run Time Average	433.8	890	6785	4.39	12.11

Average Angle of Attack	-2.55	.0573	.33	.07	.78

In NEAT optimization the evolutionary process makes it possible to graph the progression in improvement by graphing the fitness of the best performing model of each generation. Examining the maximal, minimal and average optimization of a degree of control may reveal causes for successful optimization. The graphs of the maximum optimization achieved at each degree of control show that the optimization happens suddenly and is not consistent improvement (Figs. 7, 8 & 9). The graphs of near average performing network saw a similar sporadic spike in improvement, but the spikes were much smaller (Figs. 10, 11 & 12). The poorest performing often had little to no improvement and stagnated over all of the generations (Figs. 13, 14, 15).

Graphs of Optimal Performance Demonstrated

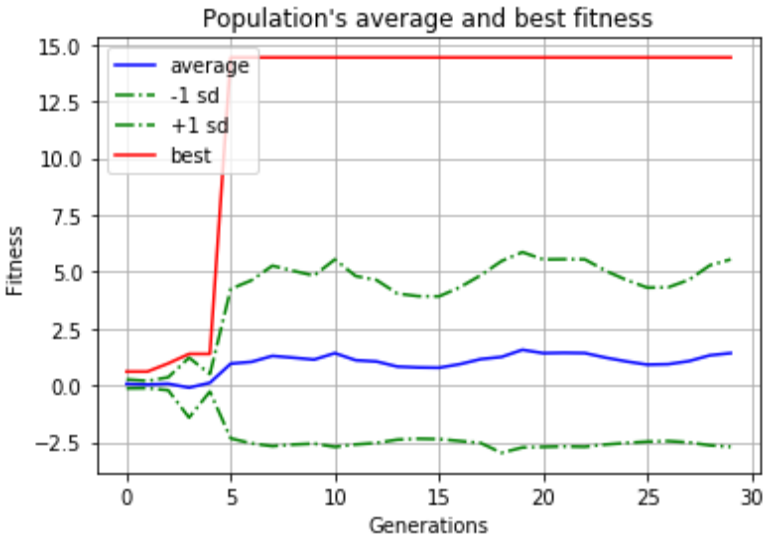

Figure 7. Graph of NEAT with one design variable, fitness in relation to the number of generations ran, optimal performance

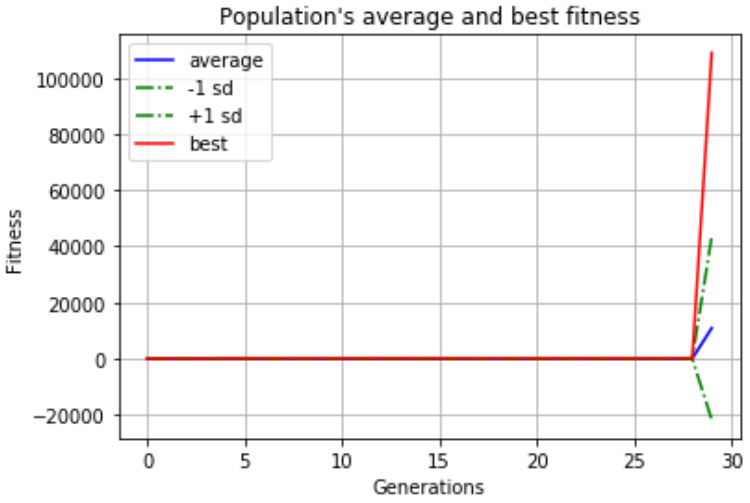

Figure 8. Graph of NEAT with six design variables, fitness in relation to the number of generations ran, optimal performance

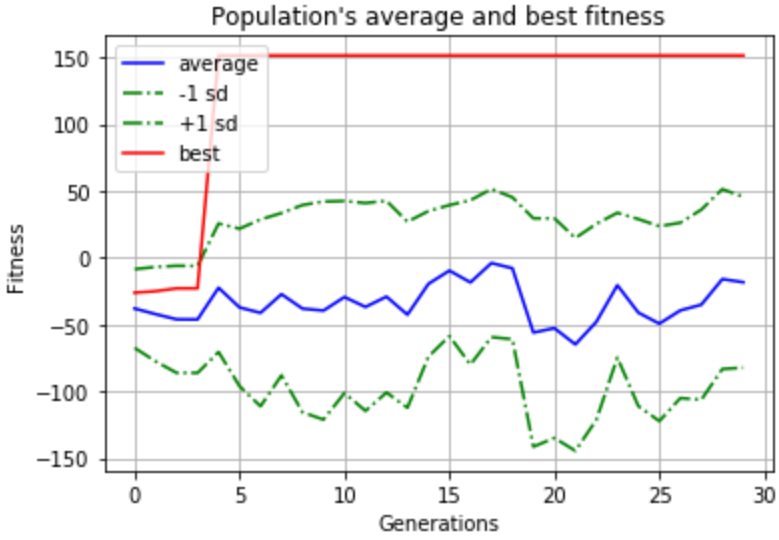

Figure 9. Graph of NEAT with 211 design variables, fitness in relation to the number of generations ran, optimal performance

Graphs of Average Performance Demonstrated

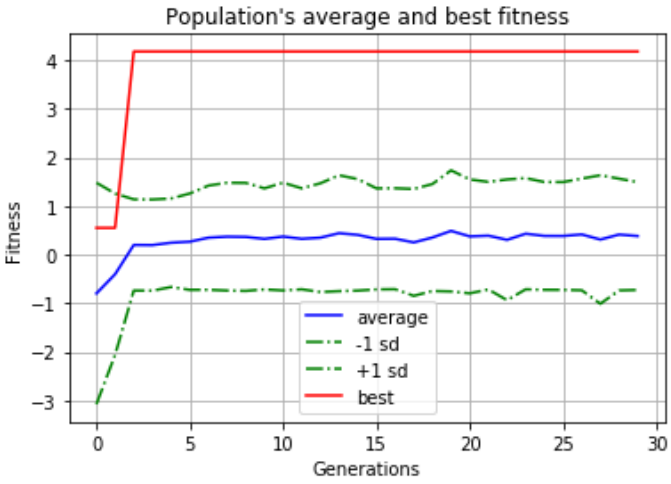

Figure 10. Graph of NEAT with one design variable, fitness in relation to the number of generations ran, average performance

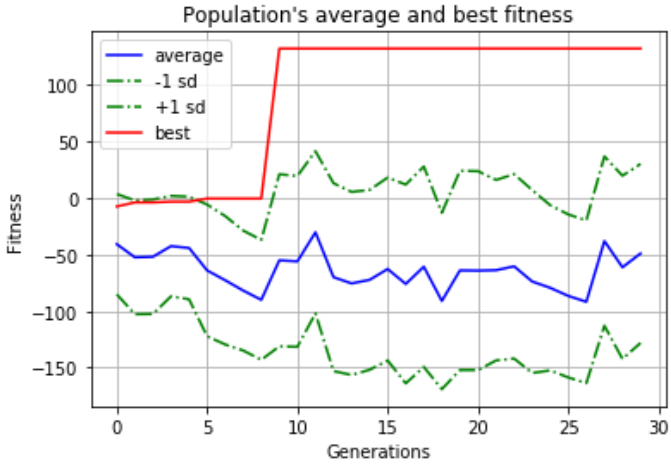

Figure 11. Graph of NEAT with six design variables, fitness in relation to the number of generations ran, average performance

Figure 12. Graph of NEAT with 211 design variables, fitness in relation to the number of generations ran, average performance

Graphs of Worst Performance Demonstrated

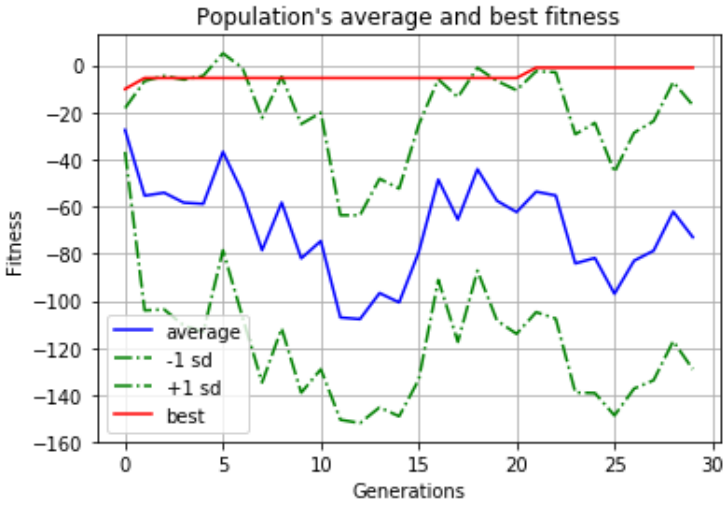

Figure 13. Graph of NEAT with six design variables, fitness in relation to the number of generations ran, worst performance

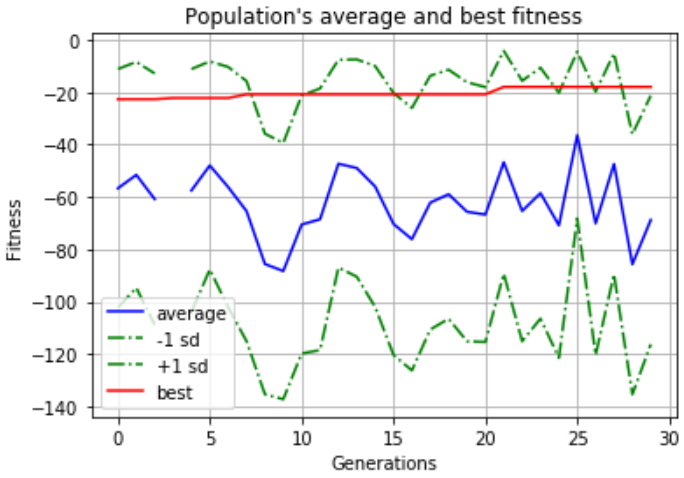

Figure 14. Graph of NEAT with 211 design variables, fitness in relation to the number of generations ran, worst performance

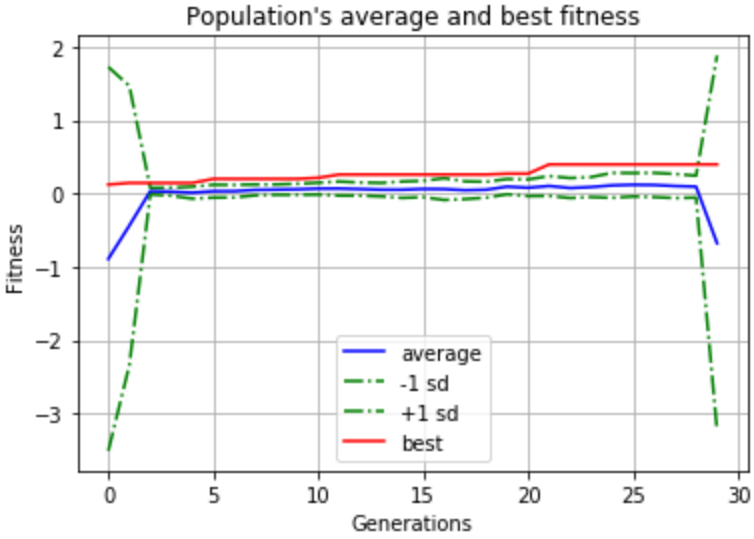

Figure 15. Graph of NEAT with one variable, fitness in relation to the number of generations ran, worst performance

For the six design variable experiments, Figure 16 is a visual representation of the initial wing design. Figure 17 is the final design after six variable MDAO optimization and its fitness is 6.23 L/D. While Figure 18 is the optimal design produced six variable NEAT Optimization, with a fitness of 109,028 L/D. Figure 19 is the initial design of the 211 variable optimization with Figure 20 being the optimal NEAT output with a fitness of 151.5 L/D.

Wings for 6 variable test

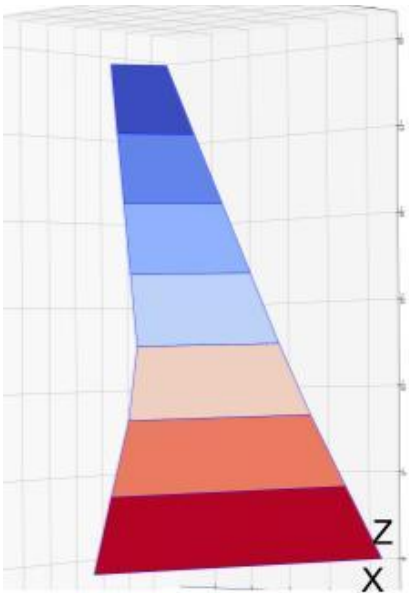

Figure 16. 6 design variable Initial wing

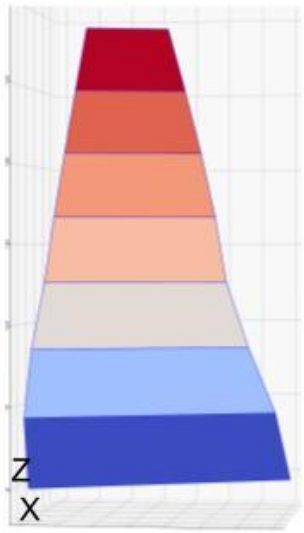

Figure 17. MDAO six design variables wing output

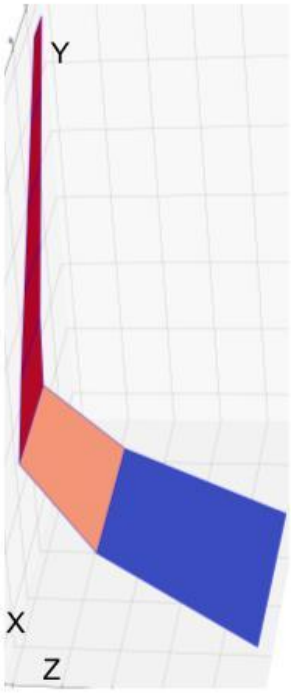

Figure 18. NEAT six design variables wing output

Wings for 211 Variable Testing

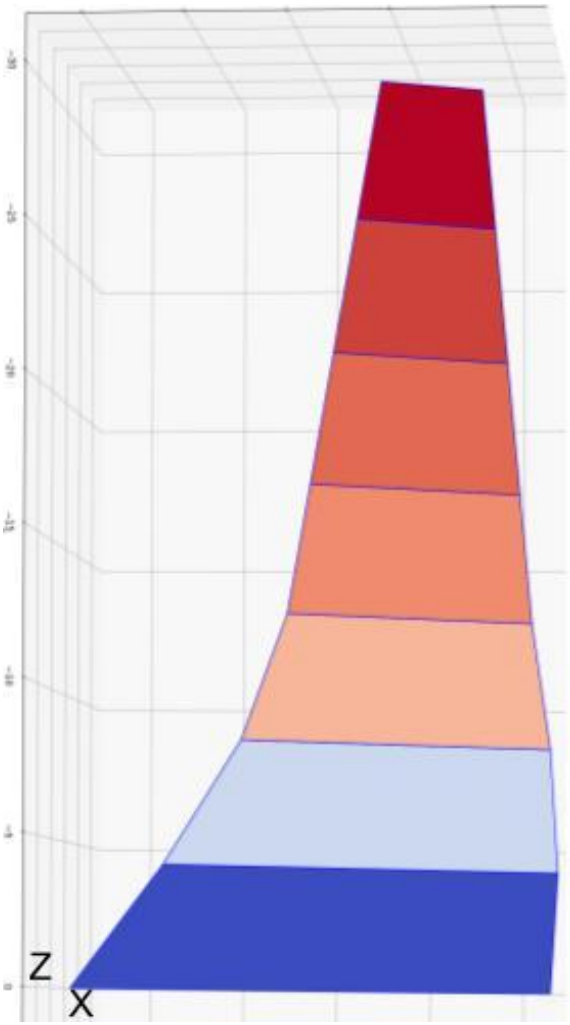

Figure 19. Initial wing for 211 design variables

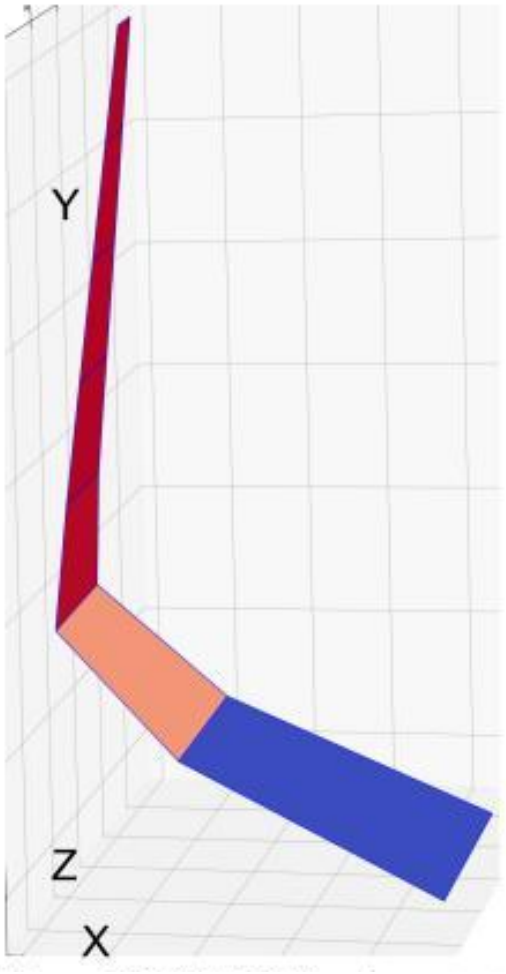

Figure 20. NEAT 211 variable design wing output

Table 2. Values from Statistical tests

	P-value
Kruskal-Wallis	0.00
Anova	.37

Table 3. The Skewness and Kurtosis of each optimization experiment

Test	Skewness	Kurtosis
Single NEAT	1.079	.36
Six NEAT	5.48	29.99
211 NEAT	2.25	4.96
Single MDAO	-1.05	-2.15
Six MDAO	30	5.48

The data results do not follow a normal distribution as most of the skewness and Kurtosis values are far from zero (Table 3). Also, visual analysis of the data plot in a histogram shows the data is not normally distributed (Fig. 21, 22, 23, & 24) . Therefore, the

Kruskal-Wallis test was used to analyze if there was any significant difference between the data sets. Furthermore, the p-value returned was 0.00 indicating with 100% confidence the null hypothesis can be rejected (Table 2). Also Figure 25 shows if there is a significant difference between two data sets with at least 95% confidence. Furthermore, by analyzing Figure 25, it is observed that there is no significant difference between: one variable Neat and 211 variable Neat, one variable Neat and 211 variable MDAO, and 211 variable Neat and one and six variable MDAO.

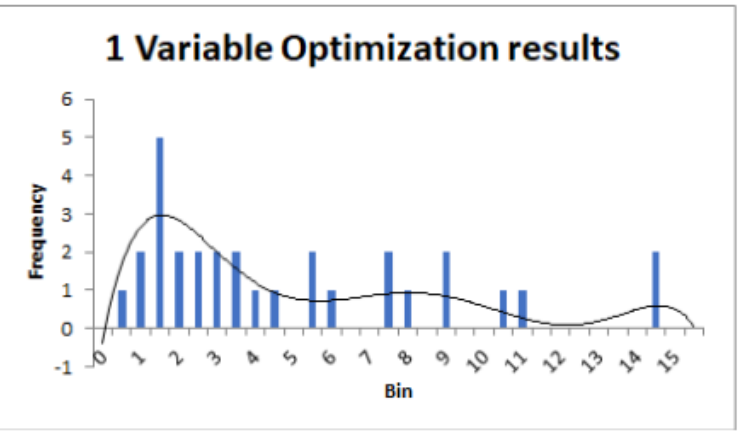

Figure 21. A historiographical representation single NEAT fitness outputs showing a non-normal distribution

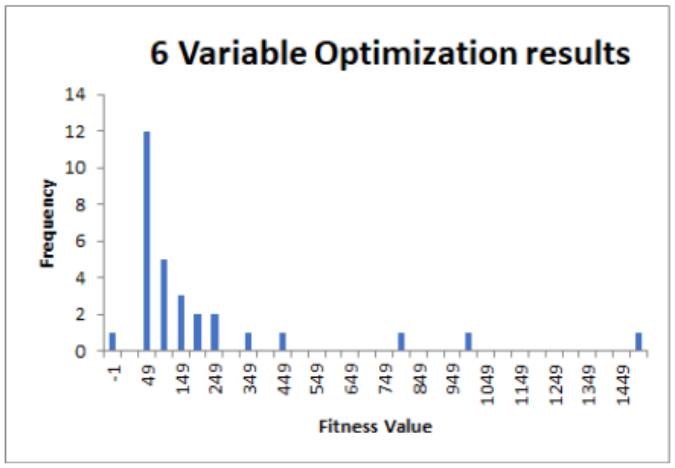

Figure 22. A historiographical representation six NEAT fitness outputs showing a non-normal distribution

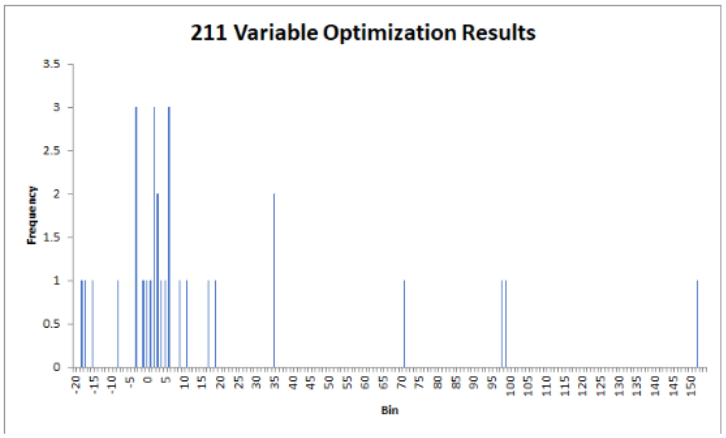

Figure 23. A historiographical representation 211 NEAT fitness outputs showing a non-normal distribution

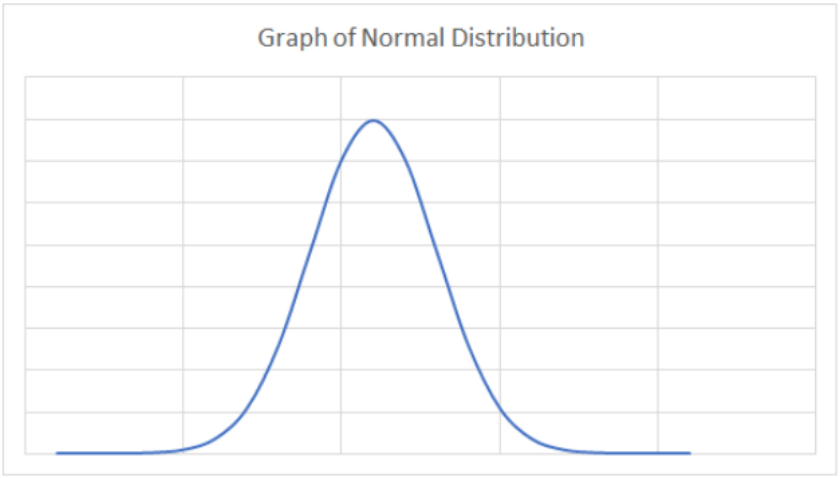

Figure 24. A graph of normal distribution for comparison

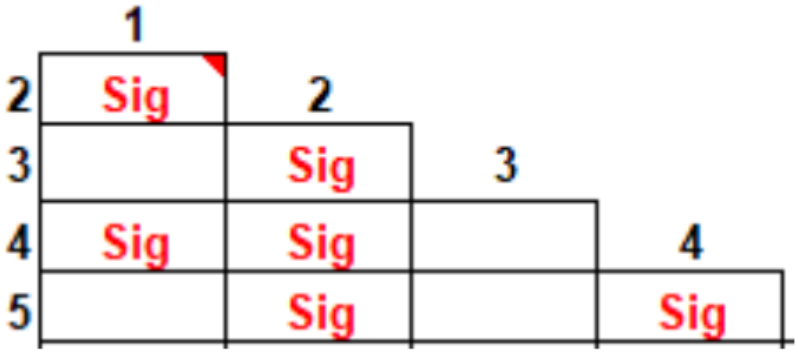

Figure 25. The Kruskal-Wallis test of significance is typically across all samples but this is a graphical representation of the results of a comparison of significance from one sample to another. One represents single NEAT, two represents Six NEAT, three represents 211 NEAT, four represents a single MDAO, and five represents a six MDAO.

Discussion

In the experiments performed, the NEAT optimization performed a higher degree of optimization than the MDAO with the same amount of control. In single variable optimization, the NEAT optimization trimmed average was 4.3 times greater than what the MDAO yielded. In an ideal case, which shows the potential of the method, the NEAT single variable was 13 times larger than the MDAO single variable. The trend continued with 6 variables of control. NEAT's trimmed average was 20.4 times greater than MDAO, and the optimally performing NEAT optimization performed 17500 times better than MDAO. The most relevant results in the simulations are the maximum values achieved as ultimately the small number of generations and population limited the possibility of a maximal optimization occurring consistently. Due to the evolutionary framework, improvement occurs randomly. With an increase in the time a simulation runs, the chance of an improvement increases. In optimization ran for an infinite

number of generations, fitness should converge on a maximal value. However, without running an optimization for an infinite or a large number of generations, it is impossible to determine what the maximal value is. However, for these network designs, the experiments determine that the maximal fitness possible is at least the maximum value achieved.

The Kruskal-Wallis test p-value indicated that there was no statistical difference between the single and 211 variable optimization. The reason for this is due to the high chance that 211 optimization will fail, leading to a large distribution of data around the same values of single variable. The 211 variable experiment results show the variability of the NEAT optimization when few generations are run. As an experiment becomes more complex, the number generations needed to a ideal optimization increases. Therefore, the average optimization of 211 variable will be small when ran over few generation. Though the average indicates that there is no different between the two degrees control, the maximal optimization achieved indicates a different conclusion.

When evaluating the effectiveness of an algorithm, the ultimate goal of the algorithm must be considered. When performing an aerodynamic optimization, the most important goal is improving the performance of an aircraft to a maximal degree. Reduction in run time is ideal because it saves money in the design phase. Since planes are long term use vehicles, saving time when performing optimization is less important than improving the aircraft design in comparison to the MDAO method of optimization. NEAT by far outperformed MDAO at optimizing aerodynamic efficiency. Though the MDAO took far less time, the time that it takes to reach a final solution is not as important as the quality of the solution. In the comparison of

the different degrees of control in NEAT, the six variable was the most effective on average and performed the most significant optimization.

Limitations

In the study, the number of hidden nodes was designed to represent an optimal NEAT network. This number was not determined using a large sample size. Therefore, the network may not be at its optimal state. An optimized performing NEAT network may have slightly improved performance over that shown in the study, but the experiments should give a good representation of how such a network would perform. Furthermore, only the NEAT-python library was used to create the NEAT algorithms used in the experiment. Other libraries may have an overall difference in efficiency, but the outcome should be similar as the fundamental design behind the algorithms is the same. Also. in the study the designs proposed had no constraints physical constraints meaning that wings produced have not been test for and probably don't have structural integrity.

Conclusion

In the comparison of the different methods of optimization, the NEAT method was overall more successful at creating efficient airplane designs. In the majority of scenarios, the design created by the NEAT network was more efficient than the product of the MDAO method.

When the network failed to create a more efficient design, it was due to lack of sufficient time to run and not a failure of the algorithm as a whole. In a scenario in which NEAT algorithms have an infinite amount of time to run, it should always reach an optimal solution. Failures in optimization, reflected insufficient time to analyze and find the ideal weights and biases for a network not a deficiency of the algorithm. When choosing the degree of control, it is crucial to consider the amount runtime allotted. The six variable NEAT seems to be an effective degree of control, but with sufficient runtime, 211 variables may achieve unseen improvements. The sporadic improvements displayed by NEAT networks in the experimentation are unavoidable due to the nature of the neuroevolutionary method. As the inspiration for neuroevolution is the process of Darwinian evolution in which rapid and successful mutations occur is random, the improvements from a NEAT network follow similar trends.

In choosing the degree of control in which a NEAT network has over an airplane design, it is critical to allow maximal influence over variables that have the greatest impact on performance. Too many variables make it difficult to optimize. For instance, the single variable optimization was successful and sometimes made significant improvements in the performance of design. Due to limited control over the entirety of the design, the degree of improvement was narrow compared to the other two degrees of control tested. The 211 variable test showed it was possible for significant improvements to occur. However, with such a large number of variables, the challenge to find optimal values hindered its consistent design improvement. Overall, the six variable optimizations were the most successful. This method had enough control over the wing design to make significant changes to the performance of a wing, but it was not burdened with trying to find optimal solutions for a large number of variables.

In determining whether to use a NEAT approach to optimize an airplane's airfoil, time and computing power allotted are the two main deciding factors. As NEAT has shown to achieve much higher levels optimization than the traditional MDAO method, it makes it the obvious choice when time and computing power are in abundance. But in a scenario of limited computing power or time, NEAT is a poor choice. In the study, relatively low-fidelity evaluations of the airfoil occurred making relatively computationally inexpensive but still the NEAT method took up to 100 times as long as the MDAO. Simulation that might normally take a couple days with MDAO could take over a year with NEAT. In unless improvements are made in the computing efficiency of NEAT, it may be inapplicable many scenarios. But the massive improvements of an airfoil in performing NEAT optimization indicate its great potential in the optimization of airplanes.

Acknowledgements

I would like to thank Mr. John Jasa, a researcher at University of Michigan Multidisciplinary Optimization Lab and the creator of OpenAerostruct for guidance with this project. Also Dr. James Brownley, USAF Statistical Researcher, was very helpful with statistical analysis. Dr. Arlsen, Aerospace Researcher CalTech, his initial guidance was very useful to the completion of this study. Most of all, Dr. Malhotra, my Internal Mentor, for her continual guidance and supporting during the research project.

References

Amari, S. (1993). Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5), 185-196. doi:10.1016/0925-2312(93)90006-o

Basheer, I., & Hajmeer, M. (2000). Artificial neural networks: Fundamentals, computing, design, and application. *Journal of Microbiological Methods*, *43*(1), 3-31. doi:10.1016/s0167-7012(00)00201-3

Berquist, J., O'brien, W., Khan, A., Tessier, A., & Attar, R. (2017). An Investigation of Generative Design for Heating, Ventilation, and Air-Conditioning. *Proceedings of the 2017 Symposium on Simulation for Architecture and Urban Design (SimAUD 2017).* doi:10.22360/simaud.2017.simaud.018

Bos, A. H. (1998). Aircraft conceptual design by genetic/gradient-guided optimization. *Engineering Applications of Artificial Intelligence, 11*(3), 377-382. doi:10.1016/s0952-1976(98)00009-8

Caldas, L. (2006). GENE_ARCH: An Evolution-Based Generative Design System for Sustainable Architecture. *Lecture Notes in Computer Science Intelligent Computing in Engineering and Architecture*, 109-118. doi:10.1007/11888598_12

Castillo, P. A., Arenas, M. G., Castillo-Valdivieso, J. J., Merelo, J. J., Prieto, A., & Romero, G. (2003). Artificial Neural Networks Design using Evolutionary Algorithms. *Advances in Soft Computing*, 43-52. doi:10.1007/978-1-4471-3744-3_5

Chen, W., Chiu, K., & Fuge, M. (2019). Aerodynamic Design Optimization and Shape Exploration using Generative Adversarial Networks. *AIAA Scitech 2019 Forum.* doi:10.2514/6.2019-2351

Chinag, C., & Youssef, H. (1994). Neural network approach to aerodynamic coefficients estimation and aircraft failure isolation design. *Guidance, Navigation, and Control Conference.* doi:10.2514/6.1994-3599

Deloach, R. (1998). Applications of modern experiment design to wind tunnel testing at NASA Langley Research Center. *36th AIAA Aerospace Sciences Meeting and Exhibit.* doi:10.2514/6.1998-713

Gulanova, J., & Veres, M. (2013). Generative Design Methods in Process of Car Body Components Development. *Machine Design, 5*(3), 121-124.

Generative Design. (n.d.). Retrieved from https://www.autodesk.com/solutions/generative-design

Gray, J., Moore, K., & Naylor, B. (2010). OpenMDAO: An Open Source Framework for Multidisciplinary Analysis and Optimization. *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*. doi:10.2514/6.2010-9101

Fernandes, R. M. (2013). Generative Design: A New Stage in the Design Process.

Floreano, D., Dürr, P., & Mattiussi, C. (2008). Neuroevolution: From architectures to learning. *Evolutionary Intelligence, 1*(1), 47-62. doi:10.1007/s12065-007-0002-4

Hochreiter, S., Younger, A. S., & Conwell, P. R. (2001). Learning to Learn Using Gradient Descent. *Artificial Neural Networks — ICANN 2001 Lecture Notes in Computer Science*, 87-94. doi:10.1007/3-540-44668-0_13

Holst, T. (2004). Genetic Algorithms Applied to Multi-Objective Aerospace Shape Optimization. *AIAA 1st Intelligent Systems Technical Conference*. doi:10.2514/6.2004-6512

Ignatyev, D., & Khrabrov, A. (2018). Experimental Study and Neural Network Modeling of Aerodynamic Characteristics of Canard Aircraft at High Angles of Attack. *Aerospace, 5*(1), 26. doi:10.3390/aerospace5010026

Jasa, J. P., Hwang, J. T., & Martins, J. R. (2018). Open-source coupled aerostructural optimization using Python. *Structural and Multidisciplinary Optimization, 57*(4), 1815-1827. doi:10.1007/s00158-018-1912-8

Krish, S. (2011). A practical generative design method. *Computer-Aided Design, 43*(1), 88-100. doi:10.1016/j.cad.2010.09.009

McIntyre, A., Kallada, M., Miguel, C. G., & Feher da Silva, C. (2017, July 3). Neat-python [Computer software]. Retrieved from Https://github.com/CodeReclaimers/neat-python

Moore, G. (1998). Cramming More Components Onto Integrated Circuits. *Proceedings of the IEEE, 86(*1), 82-85. doi:10.1109/jproc.1998.658762

Pitts, W. (1942). Some observations on the simple neuron circuit. *The Bulletin of Mathematical Biophysics, 4*(3), 121-129. doi:10.1007/bf02477942

Rai, M. (2004). Robust Optimal Aerodynamic Design Using Evolutionary Methods and Neural Networks. *42nd AIAA Aerospace Sciences Meeting and Exhibit*. doi:10.2514/6.2004-778

Ray, T., Tai, K., & Seow, K. C. (2001). Multiobjective Design Optimization By An Evolutionary Algorithm. *Engineering Optimization*, 33(4), 399-424. doi:10.1080/03052150108940926

Timnak, N., Jahangirian, A., & Seyyedsalehi, S. A. (2017). An Optimum Neural Network for Evolutionary Aerodynamic Shape Design. *Scientia Iranica, 0*(0), 0-0. doi:10.24200/sci.2017.4308

Secco, N. R., & Mattos, B. S. (2015). Artificial Neural Networks Applied to Airplane Design. *53rd AIAA Aerospace Sciences Meeting*. doi:10.2514/6.2015-1013

Shopova, E. G., & Vaklieva-Bancheva, N. G. (2006). BASIC—A genetic algorithm for engineering problems solution. *Computers & Chemical Engineering*, *30*(8), 1293-1309. doi:10.1016/j.compchemeng.2006.03.003

Sobieszczanski-Sobieski, J., & Haftka, R. (1996). Multidisciplinary aerospace design optimization - Survey of recent developments. 34th Aerospace Sciences Meeting and Exhibit. doi:10.2514/6.1996-711

Smetana, F. O., Summey, D. C., Smith, N. S., & Carden, R. K. (1975). Light aircraft lift, drag, and moment prediction: A review and analysis.

Stack, J. (1935). Report no. 492, Tests of 16 related airfoils at high speeds. *Journal of the Franklin Institute, 219*(1), 122. doi:10.1016/s0016-0032(35)91463-6

Stanley, K. (2004). Efficient Evolution of Neural Networks Through Complexification.

Stanley, K., & Miikkulainen, R. (2002). Efficient evolution of neural network topologies. *Proceedings of the 2002 Congress on Evolutionary Computation. CEC02 (Cat. No.02TH8600)*. doi:10.1109/cec.2002.1004508

Veldhuizen, D. A., & Lamont, G. B. (2000). Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. *Evolutionary Computation, 8*(2), 125-147. doi:10.1162/106365600568158

Wallach, R., Mattos, B., Girardi, R., & Curvo, M. (2006). Aerodynamic Coefficient Prediction of Transport Aircraft Using Neural Network. *44th AIAA Aerospace Sciences Meeting and Exhibit*. doi:10.2514/6.2006-658

Zhang, Y., Sung, W. J., & Mavris, D. N. (2018). Application of Convolutional Neural Network to Predict Airfoil Lift Coefficient. *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. doi:10.2514/6.2018-1903

Zhu, Q., Qin, A., Suganthan, P., & Huang, G. (2005). Evolutionary extreme learning machine. *Pattern Recognition*, *38*(10), 1759-1763. doi:10.1016/j.patcog.2005.03.028

Appendix A.

Constant parameters in all NEAT experiments

fitness_criterion = max

fitness_threshold = inf

pop_size = 10

reset_on_extinction = True

activation_default = inv

activation_mutate_rate = 0.0

activation_options = inv

aggregation_default = sum

aggregation_mutate_rate = 0.0

aggregation_options = sum

bias_init_mean = 0.0

bias_init_stdev = 1.0

bias_max_value = 1000.0

bias_min_value = -1000.0

bias_mutate_power = 0.5

bias_mutate_rate = 0.1

bias_replace_rate = 0.1

compatibility_disjoint_coefficient = 1.0

compatibility_weight_coefficient = 0.5

conn_add_prob = 0.5

conn_delete_prob = 0.5

enabled_default = True

enabled_mutate_rate = 0.10

feed_forward = True

initial_connection = full_direct

node_add_prob = 0.2

node_delete_prob = 0.2

response_init_mean = 1.0

response_init_stdev = 0.0

response_max_value = 30.0

response_min_value = -30.0

response_mutate_power = 2.0

response_mutate_rate = 0.2

response_replace_rate = 0.0

weight_init_mean = 0.0

weight_init_stdev = 1.0

Weight_mutate_power = 5.0

weight_max_value = 30

weight_min_value = -30

weight_mutate_rate = 0.5

weight_replace_rate = 0.01

compatibility_threshold = 3.0

species_fitness_func = max

max_stagnation = 3

species_elitism = 1

elitism = 1

survival_threshold = .8