IPN Progress Report 42-234 • August 15, 2023

Design of 3D Antenna Geometries Using Genetic Algorithms

Julie Rolla,* Bryan Reynolds,† Jacob Weiler,† Amy Connolly,† Ryan Debolt,† Alex Machtay,† Ben Sipe,† and Dylan Wells†

ABSTRACT. — This report presents initial results apropos a genetic algorithm that evolves 3D geometries; this early study is a proof of concept, presenting an algorithm that evolves to a predefined target shape, with the final goal of designing optimized 3D antenna geometries. The algorithm presented in this work builds structures by combining building blocks of different geometric primitives, where the fitness of a design is measured by the similarity to the target shape. Multiple techniques for comparing 3D shapes are explored and compared in the context of a fitness score for evolutionary algorithms. The designed algorithm was capable of evolving to biconical and dipole antenna shapes rapidly using a variety of fitness functions. A more complex log-periodic antenna shape was evolved using a directed fitness function comparing the component shapes. The development of this algorithm preludes incorporating more complex fitness functions that build designs to improve sensitivity to science outcomes. Future improvements to the algorithm and the steps required to achieve designs with improved sensitivity are discussed in the context of antennas but could be applied to other applications.

I. Introduction

A. Motivation

Astrophysics science experiments are often limited by the sensitivity of their detector elements, which have numerous constraints and requirements. Sensitivity in the field of radio observations can be improve with optimized antenna parameters and reduced

*Tracking System and Applications Section.

†Department of Physics, Center for Cosmology and AstroParticle Physics, The Ohio State University

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

system noise. This investigation presents an initial step in the aim to use genetic algorithms (GAs) to optimize antennas, either by improving sensitivity to science outcomes and/or reducing resource requirements (mass, volume, and cost). This report presents the initial step by demonstrating a GA that evolves 3D geometries to an existing target shape. GAs are a computational heuristic that utilize principles of evolution to efficiently identify solutions to defined problems. The focus of this investigation is on scientific remote sensing applications, such as receivers for signals of opportunity like Global Navigation Satellite System (GNSS) antennas, passive sounding, and low-frequency radio signals including radio emissions from extrasolar planets [1, 2, 3], cosmic ray electrons and cosmic magnetic fields [4, 5, 6, 7, 8], and the highly redshifted neutral hydrogen hyperfine line [9, 10, 11, 12].

This report presents the initial step in antenna evolution to science outcomes by demonstrating a GA that evolves 3D geometries to an existing target shape *without* predefined antenna types (i.e., dipole or biconical) using shape comparison fitness functions. Future iterations will evolve toward complex science goals by incorporating fitness functions related to science outcomes through integration with antenna and experiment simulation software. Antennas with a predefined geometry have previously been designed using evolutionary algorithms with limited parameters. NASA has used GAs to design unique wire antennas for the ST5 mission [13]. The Genetically Evolving Neutrino Telescopes (GENETIS) collaboration has developed GAs integrated with antenna and science simulation software to design biconical and horn antennas with improved sensitivity to neutrino signals [14, 15, 16]. Both these examples required strict constraints to predefined antenna types such as wire, biconical, and horn.

The use of GAs to aid in design of various detectors and experiments has become more prevalent in recent years [17, 18]. McCarthy et al. designed a horn antenna for the detection of Cosmic Microwave Background radiation using a GA [19]. The Long-Baseline Neutrino Oscillation (LBNO) experiment and the Deep Underground Neutrino Experiment (DUNE) optimized the design of neutrino beamlines through a GA using science simulations to determine the fitness [20, 21]. GAs have also been used to optimize various aspects of detector design, including layout, sensors, shielding, and trigger optimization [22, 23, 24, 25, 26].

B. Genetic Algorithms

GAs are a computational technique capable of finding high-quality solutions to complex problems [27, 28]. They are especially effective in problems with many parameters of high cardinality, which can be impractical to optimize with traditional techniques. GAs operate with the principles of biological evolution, with many potential solutions iteratively evolving toward improved results. The basic procedure and the nomenclature are described in this section.

GAs begin by generating an initial set (population) of potential solutions (individuals), defined by genes, which are the parameters that fully describe a solution. Each individual is evaluated by a mathematical fitness function and receives a fitness score, which reflects how well an individual performs according to the predefined objective. Individuals are chosen as parents through selection methods, and their genes are combined to produce offspring through genetic operators. As with biological evolution, individuals with higher fitness scores are more likely to be chosen as parents, and offspring are created with various methods of combining parent genes, mutating of parent genes, or introducing new individuals. Over many generations, evolutionary pressure encourages the continuation of individuals with higher fitness scores. The evolution terminates after predefined criteria are met, such as reaching a particular fitness score or number of generations.

The success of a GA is dependent on a variety of additional complexities and hyperparameters. It is important to keep in mind that GAs are a probability-based heuristic, which means that complex solutions require a large number of individuals over many generations for satisfactory results. While increasing the number of individuals and generations will improve results, this must be balanced against the accompanying increase in computation time. Additionally, the types of selection methods and operators used to generate offspring and the ratios of different types can affect the results. Some selection methods strongly favor higher fitness score individuals, which can result in the GA being stuck in a local maxima. Other selection methods and operators promote genetic diversity in parents and offspring, but may inhibit the speed and ability in finding solutions. Finally, it is crucial that the fitness function is well-defined. As will be discussed in Section IV, different fitness functions with similar goals can have significantly different results.

C. 3D Object Generation

Computational design of complex 3D objects with GAs presents unique challenges that must be considered. Each individual must be fully described by a set of genes that can be inherited by future individuals. For a single 3D object, this is relatively trivial. A cuboid, for example, is described by five genes: length, width, height, θ rotation, and ϕ rotation. However, determining the genes of a combination of shapes is significantly more complex, especially when the target shape is detailed and variable. Finding a technique that uses a single set of genes to describe 3D objects and constructing a fitness evaluation were key components of this investigation. These challenges are nontrivial and multiple methods were considered. A number of techniques have been used in literature for generating a variety of 3D objects including spherical harmonics [29], procedural modeling [30, 31, 32, 33], and voxels [34, 35, 36].

Each method was reviewed in designing this GA. Objects generated with spherical harmonics often do not have sharp edges or empty space, which is desired in our designs. Procedural modeling typically involves repeating specific geometries, and we did not want to limit the designs to those constraints. Voxel algorithms can generate complex structures out of voxels in a 3D grid that can be enabled or disabled; however, evolving an array of voxels presents a significant computational challenge due to the very large parameter space resulting from the number of genes required to describe complex objects at high resolution. In an effort to overcome this computational burden, Funes and Pollack developed a GA that builds 3D objects using combinations of cuboids, similar to "LEGO" brick structures [37]. In their algorithm, the genes for each individual consist of simple dimensional definitions (length, width, height) and connections (which cuboids the individual is connected to and where they connect). An individual is represented by a tree of shapes, each with their own genes. The algorithm presented in this report is based on an expansion of this technique.

This report details the initial results of the GA's ability to construct structures and evolve individuals to match various target shapes. Section II describes a detailed overview of the GA. Section II.A presents the procedure for generating individuals and discusses the related challenges and solutions. In Section III, we describe the process of calculating various fitness scores that were tested. The results for evolving the chosen target shapes are given in Section IV. Finally, Section V discusses the implications of the results, the next steps in generating antenna designs that evolve toward specified science outcomes, and potential challenges.

II. Genetic Algorithm

A flowchart of the GA used in this analysis is given in Figure 1. The GA has a number of different parameters outlined in Appendix I. The algorithm starts by generating an initial population of designs. Using the Python API, the designs are built in Blender [38], an open-source 3D modeling software, for fitness score evaluation. Fitness scores are calculated by comparing each individual to a target geometric structure. The fitness scores are then used to select parents and create offspring. The loop continues with the new generation evaluated in Blender as before. This process is repeated for a fixed number of generations or until the fitness score plateaus, after which the algorithm is terminated. In this section, details of each step of the GA are described, starting with a discussion of how an individual is built by genes.

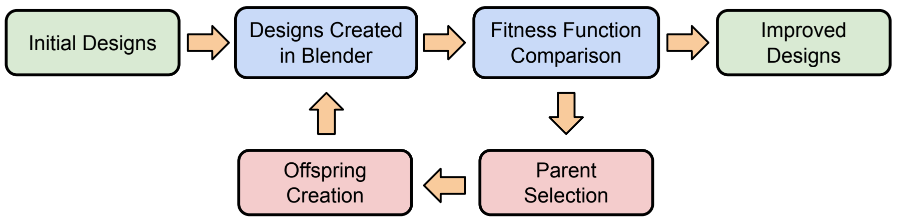


Figure 1. Diagram illustrating the basic loop of the GA. Green boxes indicate initialization and termination. The blue boxes show the steps required to evaluate the fitness function, and the red boxes give the steps for creating the next generation.

A. Building an Individual

A single individual is a combination of basic geometric primitives connected like "LEGO" building blocks to produce a more complex structure, as illustrated in Figure 2. Our algorithm begins by placing an initial base building block centered at the origin. After the base shape is placed, new building blocks can be connected to the base shape. Each new shape added now becomes a canvas for more potential building blocks to be added. The building blocks can be one of four primitive shapes: cuboids, cylinders, cones, or spheres.

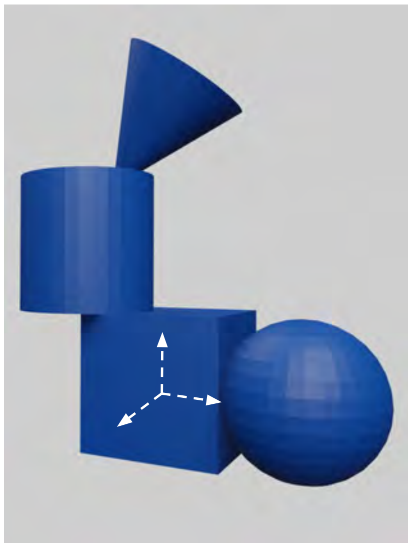


Figure 2. Example individual consisting of each of the four primitive shapes. A base cuboid at the origin with an attached cylinder and sphere. The cylinder then has an attached cone.

To describe a complete individual, the dimensions, location, orientation, and connections of each component shape are needed. These genes of a single building block are detailed in Table 1. An individual is represented by a tree structure that contains the genes of all the component shapes.

For demonstrative purposes, consider the example shape in Figure 2. This individual is built from a base cuboid, with a sphere attached on the right side, and a cylinder on the top. Above the cylinder, a cone is connected at an angle. The corresponding diagram in Figure 3 depicts the tree structure and the genes of each shape.

There are a number of complications to the description above that should be noted. First, at this initial stage of development, building blocks are only allowed to have one shape attached to any side. All shape types behave consistently, with a maximum of one attachment allowed in each region corresponding to its front, back, left, right, top, and bottom. Additionally, when cylinders and cones are added, they always connect with their flat top or bottom face.

Table 1. Primitive shape genes

Gene	Values
Shape Type	Cuboid, Cylinder, Cone, Sphere
Dimensions	Varies by shape type (See Table 2)
Location	Cartesian coordinates of shape midpoint
Rotation	θ, ϕ
Connected From	Shape built from
Connected To	Side that shapes are attached to

Table 2. Dimensions for each shape type

Shape Type	Dimension Genes
Cuboid	Length, Width, Height
Cylinder	Radius, Height
Cone	Inner Radius, Outer Radius, Height
Sphere	Radius

B. Creating Designs in Blender

Since our fitness functions depend on the comparison of individuals to a target geometry, Blender is used to 3D model individuals from their tree dictionaries in a two-step process. This integration is required as having a 3D model is critical in comparing the shape to a target (for most fitness functions). The structures are built with the following procedure. First, each building block shape is modeled and placed in 3D space. Second, the overlapping volume between the building blocks is removed and the shapes are merged together into a single mesh. As a note, for some fitness functions explored in this report, it is not necessary to remove the overlapping volume. When required, removing overlap is the largest contributor to computation time in the loop, with the other elements running in under a second. Appendix III describes efforts to minimize computation time contributed by Blender and shows examples quantifying expected computation time.

C. Fitness Function Calculation

Multiple fitness functions were developed and tested during this analysis: Hausdorff distance, average minimum distance, volume matching, linear combination, and direct dictionary comparison. These fitness functions are described in detail in Section III.

D. Parent Selection

This analysis used tournament, roulette, and rank selection methods. In tournament selection, a small subset of the population is chosen, and the individual with the

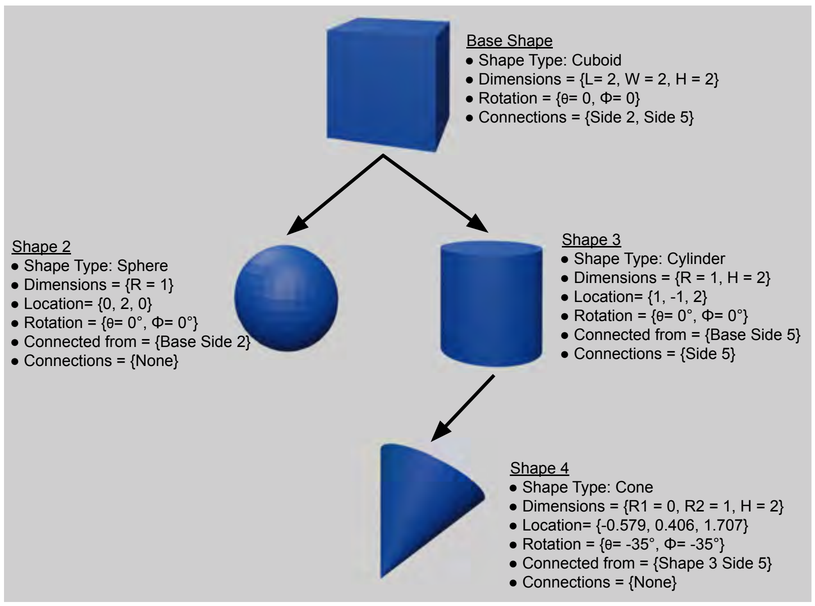

Base Shape
● Shape Type: Cuboid
● Dimensions = {L= 2, W = 2, H = 2}
● Rotation = {θ= 0, Φ= 0}
● Connections = {Side 2, Side 5}
Shape 2
● Shape Type: Sphere
● Dimensions = {R = 1}
● Location= {0, 2, 0}
● Rotation = {θ= 0°, Φ= 0°}
● Connected from = {Base Side 2}
● Connections = {None}
Shape 3
● Shape Type: Cylinder
● Dimensions = {R = 1, H = 2}
● Location= {1, -1, 2}
● Rotation = {θ= 0°, Φ= 0°}
● Connected from = {Base Side 5}
● Connections = {Side 5}
Shape 4
● Shape Type: Cone
● Dimensions = {R1 = 0, R2 = 1, H = 2}
● Location= {-0.579, 0.406, 1.707}
● Rotation = {θ= -35°, Φ= -35°}
● Connected from = {Shape 3 Side 5}
● Connections = {None}

Figure 3. Tree structure of example individual, with the genes of each shape listed. The base shape would be considered to be at a tree depth of zero, and the cone at a tree depth of 2.

highest fitness score is selected as a parent [39, 40]. In roulette, or fitness-proportionate selection, the probability of an individual being selected as a parent is proportional to their fitness score [41]. Rank selection is similar to roulette; however, instead of the selection probability being proportional to the fitness score, it is proportional to the rank of the individual.

E. Genetic Operators

Four main genetic operators were used to construct the next generation: mutation, crossover, reproduction, and injection.

Mutation is an operator that alters one gene of a single parent individual to produce an offspring. Due to the complexity of combining shapes into individuals, there are two different types of mutation used in this algorithm: (1) standard mutation, where individual genes are altered, and (2) regenerative mutation, where a shape is regenerated or altered.

Standard mutation is when either a single dimension, rotation, or location gene of a single shape is altered. For these gene mutations, the new gene is found by generating a value from a Gaussian distribution centered on the original value, where the standard deviation is a predefined percentage of the total range of values.

While standard mutation makes minor changes to individuals, regenerative mutation allows for larger changes to occur. There are three types of regenerative mutation: grow, prune, and replacement. For grow mutation, an empty side of one shape on the individual is selected and a new shape is generated that connects to that side. Conversely, with prune mutation, a shape, and all subsequent connected shapes in the tree, are removed from the individual. Finally, replacement mutation is the process of exchanging a shape with another. The new shape is generated from scratch.

At the conclusion of any type of mutation, the algorithm determines the location, rotation, and connections for each shape in the individual and alters those parameters if needed.

Crossover is a technique used to mix the genes of two parents to produce two offspring. There are two types of crossover used in this analysis: gene and branch. In gene crossover, single genes from the same shape on different parents are exchanged, such as swapping a cylinder diameter gene. Two parents that have at least one shared shape type are required. Branch crossover exchanges all shapes in one branch of the tree structure. The algorithm selects a tree depth that exists in both parents, then selects one branch from each individual to exchange. As with mutation, the algorithm ensures that the shape is still valid and that constraints are still satisfied.

The reproduction operator chooses a single parent and passes that individual to the next generation with no changes.

Injection is the creation of a brand new individual in the next generation. Individuals made by injection are created using the same method as described for generating individuals in the first generation.

III. Fitness Function Details

This section details the different fitness functions that were explored in this report.

A. Hausdorff Distance

The Hausdorff distance is the largest distance from the individual object to the closest point in the target object [42]. The Hausdorff distance, H, can be represented by Equation 1:

$$H(I,T) = \max\left[\min(\mathrm{dist}(\mathrm{I_i},\mathrm{T_j})\right], \text{ for all i and j} \tag{1}$$

where I is the individual composed of i vertices, T is the target shape composed of j vertices, and dist is the distance between two points. In effect, the Hausdorff distance gives a single value of the distance between the least correct point of the individual to the target. A Hausdorff distance of zero corresponds to identical shapes.

The fitness score, F, is calculated using the Hausdorff distance, normalized so that a perfect match is equal to 1 and decreases with imperfect matches as the Hausdorff distance increases:

$$F_H = \frac{1}{1 - H} \tag{2}$$

B. Average Minimum Distance

One new fitness function was devised that takes the average of the minimum distances from each vertex of the individual to the target. This is effectively the Hausdorff function, but instead of taking the maximum value, the mean is taken:

$$H_{\text{avg}}(I, T) = \text{mean}\left[\min(\text{dist}(\mathrm{I_i}, \mathrm{T_j})\right], \text{ for all i and j} \tag{3}$$

$$F_{H_{\text{avg}}} = \frac{1}{1 - H_{\text{avg}}} \tag{4}$$

C. Volume Matching

The volume matching fitness function compares the difference in volume between the target and an individual. Blender is used to calculate the volume of the target, V_T, the volume of the individual that lies inside the target shape, V_{in}, and the volume of the individual that lies outside of the target shape, V_{out}.

To build a fitness function, we need a value that goes to zero when the individual perfectly matches the target, and increases as either the inner volume decreases or the outer volume increases. This can be achieved with the volume score V_S:

$$V_S = |V_{\text{in}} - V_T| + V_{\text{out}} \tag{5}$$

The fitness score is then given in Equation 6:

$$F_V = \frac{1}{1 - V_S} \tag{6}$$

D. Linear Combination

The linear combination of other fitness functions was also investigated:

$$F_W = aF_H + bF_{H_{\text{avg}}} + cF_V \tag{7}$$

where a, b, and c are weights normalized to add up to one. This combination of parameters allows the GA to take advantage of the different evolutionary pressures from different fitness functions. In this report, only an equal weight of the Hausdorff distance and the average minimum distance was used, although other combinations and weights could be investigated in the future.

E. Direct Dictionary Comparison

The direct dictionary comparison fitness function sequentially evolves to a known target by increasing the fitness score of individuals whose genes are similar to that of the target shape. This fitness function forces the evolution without ambiguity, and it is only feasible when the outcome is exactly known. We developed this fitness function as an exploration into the potential of the GA in building more complex shapes, since the other fitness functions were unable to converge. This function should allow for a better understanding of the number of generations the algorithm needs to achieve a desired solution. Moving forward, this fitness function will act as a baseline for more complex shapes and will be used for continued study of hyperparameter optimization.

The direct dictionary comparison function evaluates each shape of an individual one at a time against the corresponding target shape, to get a shape sub-score:

$$S_i = \begin{cases} 0 & \text{if location or type incorrect} \\ \frac{1}{n}\sum_g^n \dfrac{1}{1+\frac{1}{M}|g_T - g_I|} & \text{if shape or type correct} \end{cases} \tag{8}$$

where i is the shape number, n is the number of genes in the shape, g_T and g_I are the value of the gene for the target and individual respectively, and M is the range of allowed values for the gene. If the shape is in the incorrect location or it is the wrong type, the sub-score is zero. If the location and type are correct, each gene, g, is compared between the target and the individual and the difference is normalized to fall between 0 and 1. The total contribution from each gene is summed, and the average is found by dividing by the number of genes, n. The final fitness score is given by:

$$F_D = \frac{\sum S_i}{N} \tag{9}$$

Note that N is the number of shapes in the target plus the number of shapes in the evolved individual that are not present in the target. This was included to reduce the fitness score of individuals with incorrect numbers of shapes.

IV. Results and Discussion

The algorithm was tested by evolving to three different target shapes: a biconical antenna, a dipole antenna, and a log-periodic antenna. For each target shape, multiple fitness functions were used and the results and convergence were compared. The direct dictionary comparison function was built for the log-periodic design and was not tested with the biconical or dipole antenna evolutions. Each generation contained 250 individuals. This section describes the details of the evolutions for each target shape and includes a discussion on insights into the different fitness functions.

A. Biconical Antenna Results

Figure 4 shows the results for a biconical target shape, or bicone, described by two connected cones with openings facing opposite directions. Evolution was tested using the Hausdorff distance, average minimum distance, volume matching, and linear combination fitness functions. Figure 4 shows that the Hausdorff distance fitness function was fastest at producing the desired shape, achieving a satisfactory result in ∼150 generations. The average minimum distance and weighted average functions both plateaued below a fitness score of 0.6, before rapidly evolving to above 0.9. The volume matching function failed to evolve to the target.

Figure 5 shows the fitness score for each individual over the first 300 generations using the Hausdorff distance fitness function. Each point represents one individual with the color corresponding to the genetic operator used to create that individual.

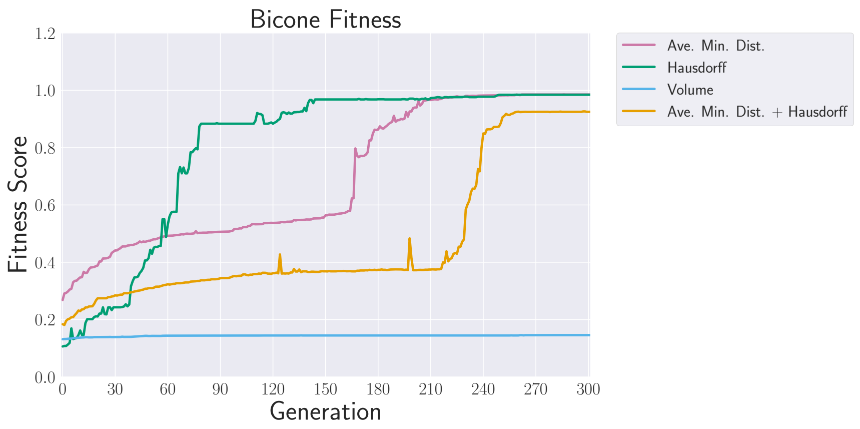


Figure 4. Comparison of the maximum fitness score per generation for the four fitness functions tested in evolving a bicone antenna over 300 generations.

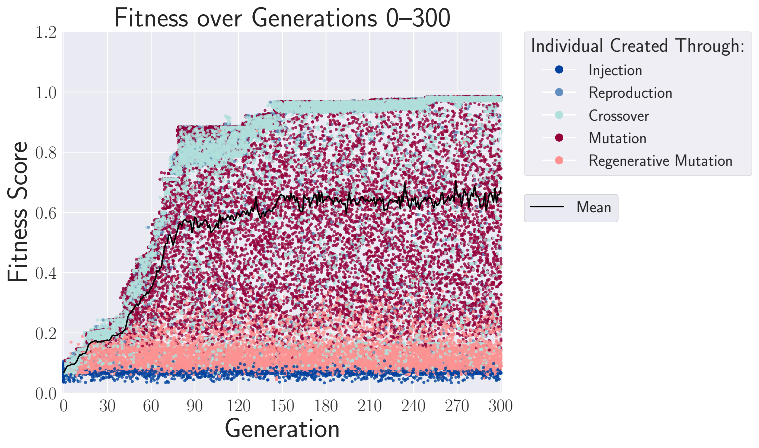


Figure 5. The fitness score of each individual of the bicone evolution using the Hausdorff distance fitness function. The ordinate gives the fitness score of that individual. The color of the point represents the genetic operator used to create that individual.

Figure 6 demonstrates the evolutionary process by showing the individual with the highest fitness score from three selected generations: the first generation, a middle generation, and the generation with the best individual. The best individual from the initial generation is a random combination of shapes. By the fiftieth generation, the best individual was comprised of two cones, but further refinement. The algorithm continued to converge to the highest scoring individual, produced in Generation 200.

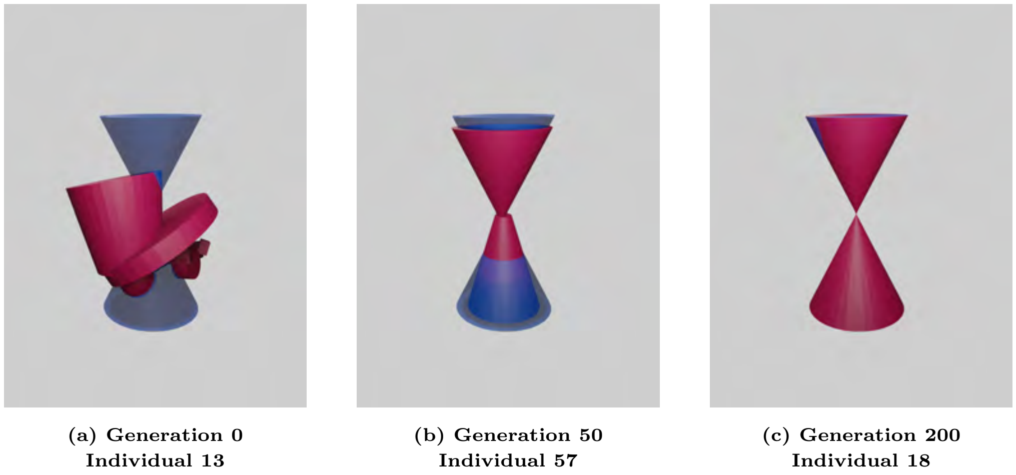


Figure 6. Three examples of the best individual of generations 0, 50, and 200 illustrating the evolution of the bicone antenna using the Hausdorff distance fitness function. The target is given in blue, and the individual is shown in red.

B. Dipole Antenna Results

Figure 7 shows the results for a dipole target shape consisting of two connected and concentric cylinders with the same diameter and height. The dipole was tested using the same fitness functions as the bicone. Figure 7 shows that the fastest convergence occurred with the linear combination fitness function.

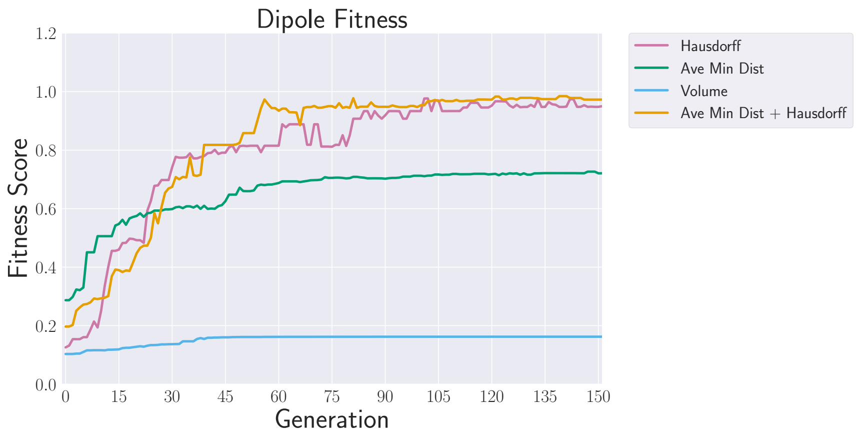


Figure 7. Comparison of the maximum fitness score per generation for the four fitness functions tested in evolving a dipole antenna over 150 generations.

The Hausdorff distance function converged at a similar rate, while the average minimum distance and volume matching functions failed to converge to the target shape. Figure 8 shows the fitness score for each individual over each generation using the weighted average of the Hausdorff distance and average minimum distance fitness functions. Figure 9 depicts the highest-scoring individuals from three generations showing the clear evolution using the linear combination fitness function.

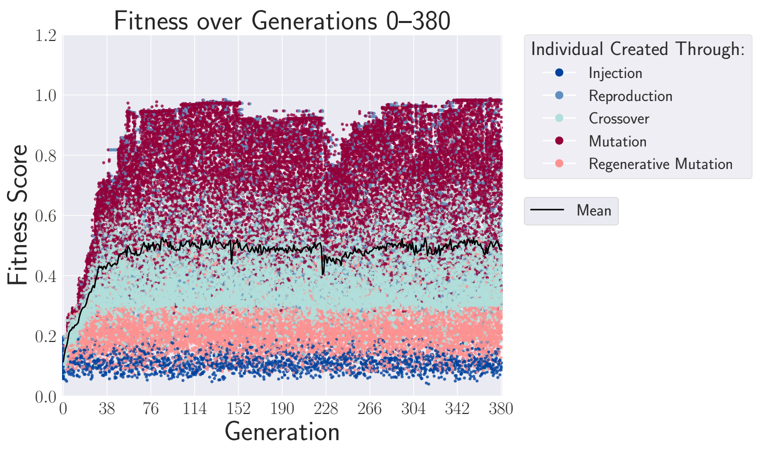
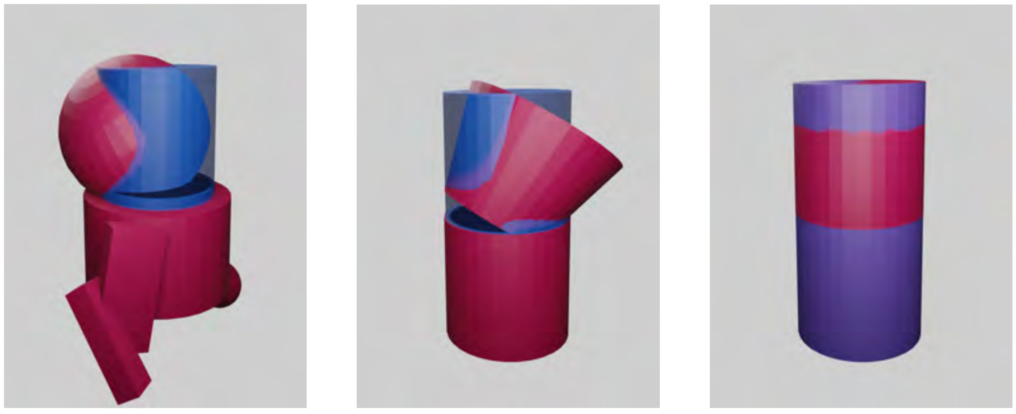


Figure 8. The fitness score of each individual of the dipole evolution using the weighted average of the Hausdorff distance and ave. min. distance fitness functions. The ordinate gives the fitness score of that individual, and the color of the point represents the genetic operator used to create that individual.



(a) Generation 0 Individual 108

(b) Generation 10 Individual 148

(c) Generation 454 Individual 51

Figure 9. Three examples of the best individual of generations 0, 10, and 454, of the dipole evolution using the average minimum distance fitness function. The target is given in blue, and the individual is shown in red.

C. Log-Periodic Antenna Results

With successful results in generating simpler antenna shapes, the fitness functions were tested on a more complicated log-periodic design. A log-periodic antenna consists of a series of dipole antennas connected perpendicularly to a support, with the dipoles decreasing in size along the length. A simple log-periodic target shape was constructed with 4 dipole arms. The target shape consisted of 14 total primitives, compared to the 2 used in the bicone and dipole tests. The results are presented in Figure 10. For this study, the fitness functions tested previously were inadequate in converging to the log-periodic shape in the tested number of generations. The Hausdorff distance fitness function failed to find multiple offshoots from the main shape, while the average minimum distance and volume matching tended to get stuck in broad approximations of the shape. The direct dictionary comparison function was consequently built to drive the evolution toward a more complex target shape.

Figure 11 shows the fitness score for each individual over each generation using the direct dictionary comparison fitness function. Figure 12 depicts the highest scoring log-periodic individuals from three generations. The run was continued until the evolved individual visually matched the target in Generation 21,000.

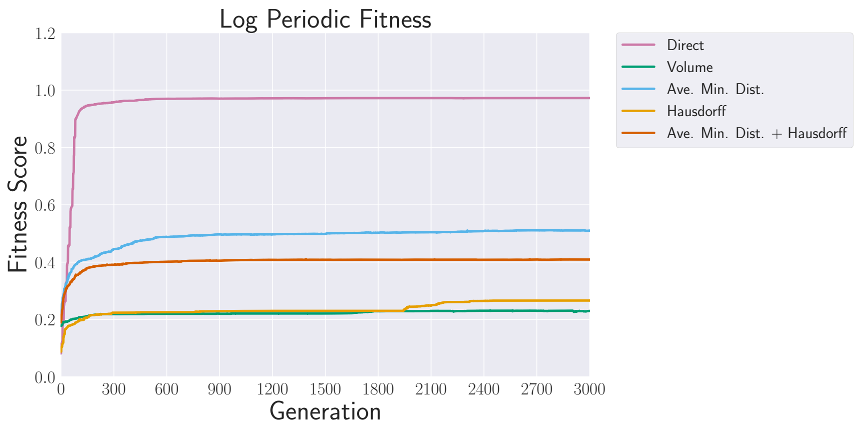


Figure 10. Comparison of the maximum fitness score per generation for the five fitness functions tested in evolving to a log-periodic antenna over 3000 generations.

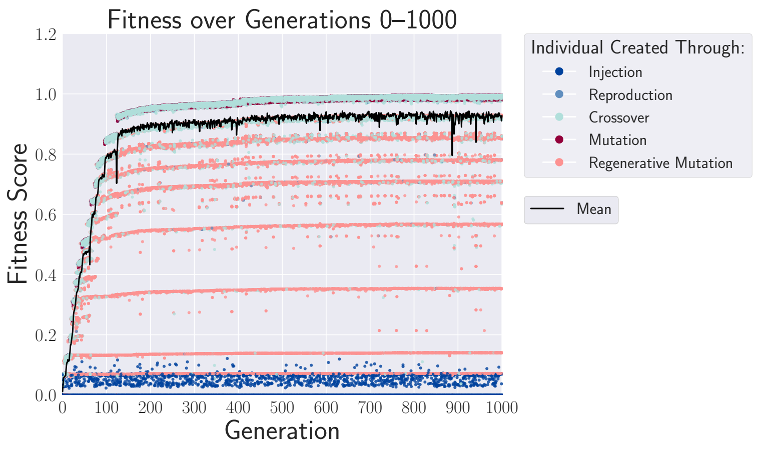
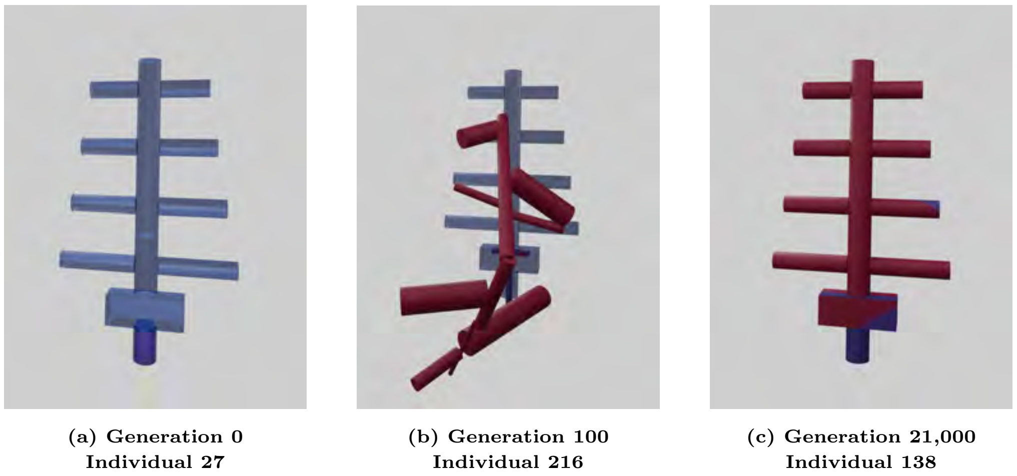


Figure 11. The fitness score of each individual of the log-periodic evolution using the direct dictionary comparison fitness function. The ordinate gives the fitness score of that individual, and the color of the point represents the genetic operator used to create that individual.


(a) Generation 0
Individual 27

(b) Generation 100
Individual 216

(c) Generation 21,000
Individual 138

Figure 12. Three examples of the best individual of generations 0, 100, and 21,000, of the log-periodic evolution using the direct dictionary comparison fitness function. The target is given in blue, and the individual is shown in red.

D. Discussion of Fitness Functions

The results presented here provide insight into the different fitness functions and the importance of choosing the right fitness function for a goal. Of particular note, GAs are stochastic processes so the efficiency and results from one evolution are unlikely to occur in subsequent runs. This can result in a wide range of convergence times, which makes comparing fitness functions difficult without performing many evolutions. This effect and a preliminary investigation are discussed in Appendix II.

The Hausdorff fitness function proved effective; however, we found that for certain geometric structures, it is not capable of adequately assessing the overall similarity between some produced structures and a target. The Hausdorff distance finds and measures the worst single point of the individual; thus, any change to a shape that is smaller than the worst point will not affect the fitness score, regardless of if it improves the individual (like adding a small object in the correct location). This is especially problematic with target shapes that have large individual protrusions. This can result in large individuals that loosely approximate the target, but do not converge since virtually any change reduces the fitness score. This issue prevents the GA from evolving to more complex shapes. The other fitness functions tested were designed to ameliorate this issue.

The volume matching function failed to converge to the target for any of the tested shapes, despite the theoretical potential as a fitness function. The volume matching function quickly found local maxima that loosely approximated the target. For example, in the dipole evolution (Figure 7), it created two stacked spheres with the approximate size of the target. While it was possible for the algorithm to generate an individual that improved on the target, the probability was extremely low, with virtually all changes resulting in a worse fitness score. The algorithm would have needed to change a sphere to a cylinder of nearly the correct size in order to improve upon its best individuals. It may be valuable to adjust this fitness function by weighting the importance of the inside and outside volume terms.

Figures 5, 8, and 11 illustrate interesting trends showing the types of individuals that are produced by the different operators. First, the injection operator consistently produces individuals with low fitness scores. This is not unexpected as these individuals are generated and therefore do not use information from previous generations to produce a solution; however, it provides new, diverse genetic information that can be used to make better designs. Regenerative mutation typically produces individuals with fitness scores below 0.3. This is crucial in the early generations, as it helps ensure that the correct shape types are represented in the population. Once the evolution approaches the target, it is likely that changing a shape will drastically hurt the individual's fitness score. Additionally, the mean fitness score increases initially before plateauing (at different fitness scores for different target shapes and fitness functions) as the diversity generated through injection balances out any improvements found through the other operators.

Individuals produced through mutation tended to have the highest fitness, while individuals made with crossover tended to exist in a midrange band. The exception to this is the bicone, where crossover and mutation produced the highest-scoring individuals. This is likely because, in the current iteration of the algorithm, crossover recalculates from scratch the connection point of the branches. This means the centers of two cylinders of the dipole (or log-periodic antenna) have a low probability of being aligned correctly, resulting in a worse fitness score. However, this was not an issue with the bicone because the connection point was the narrow ends of the cones, which left less room for error in placement. In the future, connection points will be retained with crossover to improve the effectiveness.

V. Conclusions

This report presents progress toward the evolution of 3D geometries for antenna designs using combinations of smaller component shapes. The algorithm constructed was effective in building a variety of antenna shapes using shape comparison fitness scores, and the results of this work illuminate the future efforts required for the design of antennas optimized for science outcomes.

The algorithm was able to evolve matches to bicone and dipole antenna shapes using multiple fitness functions. The evolution of a log-periodic antenna was more complex and required a specific fitness function to achieve a matching result. This work allowed for a deeper understanding of shape comparison fitness functions and their pitfalls in the context of genetic algorithms.

While significant effort and discussion went into fitness scores for shape comparisons to prove the efficacy of the algorithm, the future of this work lies in building fitness functions tied to science outcomes and improving the algorithm to make the design of optimized antennas more efficient.

A. Future Work

The main focus of future work will be to incorporate antenna gain patterns to fitness functions using XFdtd (Remcom) [43]. The ultimate goal will be using the XFdtd simulation to target specific gain patterns or science simulation outcomes, which will allow the design of unique antennas that deliver optimized sensitivity with unique constraints. This will increase computation time and require efficiency improvements to the algorithm.

There are a number of improvements to the GA that can be made by increasing the complexity of individuals. First, the algorithm currently only allows one shape to be attached per side. This constraint was made to simplify early versions of the algorithm, but it limits design opportunities. Also, new primitive shapes could be added such as pyramids, triangular prisms, trapezoids, toruses, and helices. Furthermore, once integration with XFdtd is completed, including a gene for the material could allow the unique evolution of dielectric antennas or antennas with metamaterials, such as those discussed in References [44, 45, 46, 47, 48].

GAs require a balance between genetic diversity and elitism when striving for efficiency toward a solution. This investigation highlighted how the value of different selection methods and genetic operators varies throughout the run. In the beginning, techniques (like regenerative mutation) that encourage significant changes are important in breaking out of highly incorrect solutions; however, at the end of the run, techniques that focus on optimizing a design through minor adjustments are most valuable. Creating dynamic selection methods and operators that allow for these techniques to change based on the stage of the run could increase convergence speed.

There are also optimizations that could help reduce computational requirements. For instance, computation time could be reduced by not recalculating fitness scores for individuals that have already been evaluated in a previous generation (such as those produced through reproduction). As shown in Appendix I, there are over 50 hyperparameters in this algorithm, and conducting tests to optimize these hyperparameters—potentially with the direct dictionary comparison fitness function—could reduce computation time by improving the algorithm's rate of convergence to solutions. Optimization can be challenging since GAs are probabilistic in nature and it may be valuable to repeat the same test many times, as shown with the preliminary study in Appendix II.

More advanced improvements to the algorithm will also be investigated, including particle swarms and other heuristic methods [49, 50, 51, 52], or neural networks [53, 54], to improve efficiency. Particle swarms could help reduce the number of ineffective individuals being generated and help to converge on a solution more quickly. A neural network could be used to predict the fitness of individuals similar to previously tested individuals and reduce the computational time needed to calculate fitness.

The work presented in this report demonstrates a first step in the optimization of 3D geometries for detector design. The algorithmic design of structures is a complex problem, and continued research in this area will allow for future optimization of designs for science outcomes.

Acknowledgments

The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004). The authors are grateful to the entire GENETIS collaboration for their prior work and continued assistance.

References

[1] W. M. Farrell, M. D. Desch, and P. Zarka, "On the possibility of coherent cyclotron emission from extrasolar planets," *JGR Planets*, vol. 104(E6), pp. 14 025–14 032, June 1999.

[2] P. Zarka, R. A. Treumann, B. P. Ryabov, and V. B. Ryabov, "Magnetically-driven planetary radio emissions and application to extrasolar planets," *APSS*, vol. 277, pp. 293–300, June 2001.

[3] T. J. W. Lazio and W. M. Farrell, "Radio detection of extrasolar planets: present and future prospects," *Planetary Radio Emissions VI*, pp. 603–610, 2006.

[4] E. Feenberg and H. Primakoff, "Interaction of cosmic-ray primaries with sunlight and starlight," *Physical Review*, vol. 73(5), pp. 449–469, March 1948.

[5] J. E. Felten and P. Morrison, "Omnidirectional inverse compton and synchrotron radiation from cosmic distributions of fast electrons and thermal photons," *Astrophysical Journal*, vol. 146, p. 686, December 1966.

[6] M. J. Rees, "Studies in radio source structure-III. Inverse Compton radiation from radio sources," *mnras*, vol. 137, p. 429, January 1967.

[7] G. R. Blumenthal and R. J. Gould, "Bremsstrahlung, synchrotron radiation, and Compton scattering of high-energy electrons traversing dilute gases," *Reviews of Modern Physics*, vol. 42(2), pp. 237–271, January 1970.

[8] L. Feretti and G. Giovannini, "Diffuse cluster radio sources (review)," *Proceedings of the 175th Symposium of the International Astronomical Union*, vol. 175, p. 333, January 1996.

[9] S. R. Furlanetto, S. P. Oh, and F. H. Briggs, "Cosmology at low frequencies: the 21 cm transition and the high-redshift Universe," *Physics Report*, vol. 433(4-6), pp. 181–301, October 2006.

[10] J. R. Pritchard and A. Loeb, "Evolution of the 21cm signal throughout cosmic history," *Physical Review D*, vol. 78(10), p. 103511, November 2008.

[11] ——, "Constraining the unexplored period between the dark ages and reionization with observations of the global 21 cm signal," *Physical Review D*, vol. 82(2), p. 023006, July 2010.

[12] A. Liu, J. R. Pritchard, M. Tegmark, and A. Loeb, "Global 21 cm signal experiments: a designer's guide," *Physical Review D*, vol. 87(4), p. 043002, February 2013.

[13] G. Hornby, A. Globus, D. Linden, and J. Lohn, "Automated antenna design with evolutionary algorithms," *AIAA Space 2006*, vol. 7242, September 2006.

[14] J. Rolla, A. Machtay, A. Patton, W. Banzhaf, A. Connolly, R. Debolt, L. Deer, E. Fahimi, E. Ferstle, P. Kuzma, C. Pfendner, B. Sipe, K. Staats, and S. A. Wissel, "Using evolutionary algorithms to design antennas with greater sensitivity to ultra high energy neutrinos," *Physical Review D*, 2023, forthcoming.

[15] J. Rolla, D. Arakaki, M. Clowdus, A. Connolly, R. Debolt, L. Deer, E. Fahimi, E. Ferstl, S. Gourapura, C. Harris, L. Letwin, A. Machtay, A. Patton, C. Pfendner, C. Sbrocco, T. Sinha, B. Sipe, K. Staats, J. Trevithick, and S. Wissel, "Evolving antennas for ultra-high energy neutrino detection," *37th International Cosmic Ray Conference*, 2021.

[16] J. Rolla, A. Connolly, K. Staats, S. Wissel, D. Arakaki, I. Best, A. Blenk, B. Clark, M. Clowdus, S. Gourapura, C. Harris, H. Hasan, L. Letwin, D. Liu, C. Pfendner, J. Potter, C. Sbrocco, T. Sinha, and J. Trevithick, "Evolving antennas for ultra-high energy neutrino detection," *36th International Cosmic Ray Conference*, 2019.

[17] B. Liu, H. Hu, H. Han, H. Lv, and L. Li, "Optimization of the design of gas cherenkov detectors for icf diagnosis," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 897, pp. 54–58, July 2018.

[18] A. Liu, A. Bross, and D. Neuffer, "Optimization of the magnetic horn for the nuSTORM non-conventional neutrino beam using the genetic algorithm," *Nuclear Inst. and Methods in Physics Research, A*, vol. 794, pp. 200–205, 2015.

[19] D. McCarthy, N. Trappe, J. A. Murphy, C. O'Sullivan, M. Gradziel, S. Doherty, P. G. Huggard, A. Polegro, and M. van der Vorst, "The optimisation, design and verification of feed horn structures for future cosmic microwave background missions," *Infrared Physics & Technology*, vol. 76, pp. 32–37, 2016.

[20] M. Calviani, S. di Luise, V. Galymov, and P. Velten, "Optimization of neutrino fluxes for future long baseline neutrino oscillation experiments," *Nuclear and Particle Physics Proceedings, 37th International Conference on High Energy Physics*, vol. 273-275, pp. 2681–2683, 2016.

[21] H. Schellman, "LBNF beamline," *39th International Conference on High Energy Physics*, vol. 340, 2018.

[22] A. G. Baydin, K. Cranmer, P. d. C. Manzano, C. Delaere, D. Derkach, J. Donini, T. Dorigo, A. Giammanco, J. Kieseler, L. Layer, G. Louppe, F. Ratnikov, G. C. Strong, M. Tosi, A. Ustyuzhanin, P. Vischia, and H. Yarar, "Toward machine learning optimization of experimental design," *Nuclear Physics News*, vol. 31(1), pp. 25–28, 2021.

[23] D. Kastanya, "ADORE-GA: genetic algorithm variant of the ADORE algorithm for ROP detector layout optimization in CANDU reactors," *Annals of Nuclear Energy*, vol. 46, pp. 160–168, 2012.

[24] E. B. Flynn and M. D. Todd, "Optimal placement of piezoelectric actuators and sensors for detecting damage in plate structures," *Journal of Intelligent Material Systems and Structures*, vol. 21(3), pp. 265–274, 2010.

[25] N. Kleedtke, M. Hua, and S. Pozzi, "Genetic algorithm optimization of tin–copper graded shielding for improved plutonium safeguards measurements." *Nuclear Inst. and Methods in Physics Research, A*, vol. 988, 2021.

[26] S. Abdullin, "Genetic algorithm for SUSY trigger optimization in CMS detector at LHC," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 502(2), pp. 693–695, 2003, Proceedings of the VIII International Workshop on Advanced Computing and Analysis Techniques in Physics Research.

[27] J. Frenzel, "Genetic algorithms," *IEEE Potentials*, vol. 12(3), pp. 21–24, 1993.

[28] K. Man, K. Tang, and S. Kwong, "Genetic algorithms: concepts and applications [in engineering design]," *IEEE Transactions on Industrial Electronics*, vol. 43(5), pp. 519–534, 1996.

[29] L. Shen, H. Farid, and M. A. McPeek, "Modeling three-dimensional morphological structures using spherical harmonics," *Evolution*, vol. 63(4), pp. 1003–1016, 2009. https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1558-5646.2008.00557.x

[30] P. Merrell and D. Manocha, "Model synthesis: a general procedural modeling algorithm," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17(6), pp. 715–728, 2011.

[31] V. Krs, R. Měch, M. Gaillard, N. Carr, and B. Benes, "Pico: procedural iterative constrained optimizer for geometric modeling," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27(10), pp. 3968–3981, 2021.

[32] M. Gaillard, V. Krs, G. Gori, R. Měch, and B. Benes, "Automatic differentiable procedural modeling," *Computer Graphics Forum*, vol. 41(2), pp. 289–307, 2022. https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14475

[33] K. Haubenwallner, H.-P. Seidel, and M. Steinberger, "Shapegenetics: Using genetic algorithms for procedural modeling," *Computer Graphics Forum*, vol. 36(2), pp. 213–223, 2017. https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13120

[34] Y. Zheng, L. Wu, X. Liu, Z. Chen, Q. Liu, and Q. Huang, "Neural volumetric mesh generator," *NeurIPS 2022 Workshop on Score-Based Methods*, 2022.

[35] M. Mehralian and B. Karasfi, "Rdcgan: Unsupervised representation learning with regularized deep convolutional generative adversarial networks," *2018 9th Conference on Artificial Intelligence and Robotics and 2nd Asia-Pacific International Symposium*, pp. 31–38, 2018.

[36] T. Moriya, "VoxelDCGAN," *GitHub*, 2017. https://github.com/maxorange/voxel-dcgan/

[37] P. Funes and J. Pollack, "Evolutionary body building: adaptive physical designs for robots," *Artificial Life*, vol. 4(4), pp. 337–357, 1998.

[38] B. O. Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. http://www.blender.org

[39] J. Zhong, X. Hu, J. Zhang, and M. Gu, "Comparison of performance between different selection strategies on simple genetic algorithms," *International Conference on Computational Intelligence for Modelling, Control and Automation and International*

Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06), vol. 2, pp. 1115–1121, 2005.

[40] A. Shuckla, H. M. Pandey, and D. Mehrotra, "Comparative review of selection techniques in genetic algorithm." In *2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*, pp. 515–519. IEEE, 2015.

[41] A. Lipowski and D. Lipowska, "Roulette-wheel selection via stochastic acceptance," *Physica A: Statistical Mechanics and its Applications*, vol. 391(6), pp. 2193–2196, 2012.

[42] R. T. Rockafellar and R. Wets, *Variational Analysis.* Berlin, Germany: Springer-Verlag, 2009.

[43] R. Luebbers, "XFDTD and beyond-from classroom to corporation," *2006 IEEE Antennas and Propagation Society International Symposium*, pp. 119–122, 2006.

[44] S. M. Hanham, T. S. Bird, A. D. Hellicar, and R. A. Minasian, "Evolved-profile dielectric rod antennas," *IEEE Transactions on Antennas and Propagation*, vol. 59(4), pp. 1113–1122, 2011.

[45] M. A. Belen and P. Mahouti, "Design of nonuniform substrate dielectric lens antennas using 3d printing technology," *Microwave and Optical Technology Letters*, vol. 62(2), pp. 756–762, 2020. https://onlinelibrary.wiley.com/doi/abs/10.1002/mop.32065

[46] M. Tam and R. D. Murch, "Half volume dielectric resonator antenna designs," *Electronics Letters*, vol. 33(23), pp. 1914–1916, 1997.

[47] R. W. Ziolkowski, P. Jin, and C.-C. Lin, "Metamaterial-inspired engineering of antennas," *Proceedings of the IEEE*, vol. 99(10), pp. 1720–1731, 2010.

[48] A. Erentok and R. W. Ziolkowski, "Metamaterial-inspired efficient electrically small antennas," *IEEE Transactions on Antennas and Propagation*, vol. 56(3), pp. 691–707, 2008.

[49] Y. Zhang, S. Wang, and G. Ji, "A comprehensive survey on particle swarm optimization algorithm and its applications," *Mathematical Problems in Engineering*, vol. 2015, p. 931256, Oct 2015. https://doi.org/10.1155/2015/931256.

[50] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: an overview," *Soft computing*, vol. 22, pp. 387–408, 2018.

[51] G. Papazoglou and P. Biskas, "Review and comparison of genetic algorithm and particle swarm optimization in the optimal power flow problem," *Energies*, vol. 16(3), p. 1152, 2023.

[52] R. Thangaraj, M. Pant, A. Abraham, and P. Bouvry, "Particle swarm optimization: hybridization perspectives and experimental illustrations," *Applied Mathematics and Computation*, vol. 217(12), pp. 5208–5226, 2011.

[53] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft computing*, vol. 9(1), pp. 3–12, 2005.

[54] F. van den Bergh and A. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8(3), pp. 225–239, 2004.

APPENDICES

I. Genetic Algorithm Parameters

Table 3. Parameters used in Genetic Algorithm

Category	Type	Parameters
General	Run	Num. Individuals, Num. Generations, Fitness Function
	Individual	Max. Tree Depth, Max. Shapes
Shape	Cuboid	Shape allowed, Length, Width, and Height Ranges
	Sphere	Shape allowed, Radius Range
	Cylinder	Shape allowed, Radius, and Height Range
	Cone	Shape allowed, Radius 1, Radius 2, and Height Range
Selection Methods	Tournament	Percent of parents, Group Size
	Roulette	Percent of parents
	Rank	Percent of parents
Genetic Operators	Dim. Mutation	Percent of children, St. Dev. %
	Rotation Mutation	Percent of children, St. Dev. %
	Location Mutation	Percent of children, St. Dev. %
	Grow Mutation	Percent of children
	Prune Mutation	Percent of children
	Regen. Mutation	Percent of children
	Gene Crossover	Percent of children
	Branch Crossover	Percent of children
	Reproduction	Percent of children
	Injection	Percent of children

II. Run Variability Analysis

When discussing results from a GA, it is important to recall that they rely on inherently stochastic processes. As such, repeated uses of the same algorithm with the same parameters, operator and selection method settings, and fitness function have no guarantee to arrive at the same solution in the same number of generations. To quantify the impact of the stochastic nature of the GAs on the efficiency of evolutions, repeated runs were conducted for the evolution to the target dipole design using both the average minimum distance and equal weighted combination of average minimum distance and Hausdorff distance fitness metrics.

For these runs, all settings of the GA were identical to those used in Section IV, with the data discussed there also included in this supplementary discussion. Each additional run was conducted for 250 generations. Each repeated run exhibits unique behavior, with large variance in the number of generations needed to reach a satisfactory solution. Some conclusions about representative behaviors may however be drawn. In the repeated runs using the average minimum distance fitness score metric, shown in Figure 13, it is seen that the most common outcome is that this metric will guide the GA to produce a non-optimal solution with a midrange fitness score, and will less frequently find an optimal solution quite quickly (under approximately 50 generations). In the repeated runs using the weighted combination of average minimum distance and Hausdorff distance as the fitness score metric, shown in Figure 14, the inverse is true. These runs more commonly achieve a near-optimal solution relatively quickly (under around 75 generations) and less frequently find only a solution with a midrange fitness score in the same number of generations.

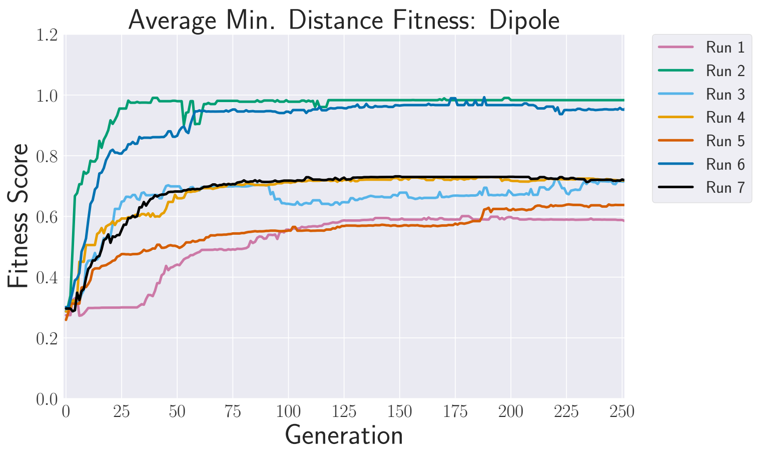


Figure 13. Results of repeated runs using the average minimum distance fitness score metric for shape-to-shape evolution to the target dipole design.

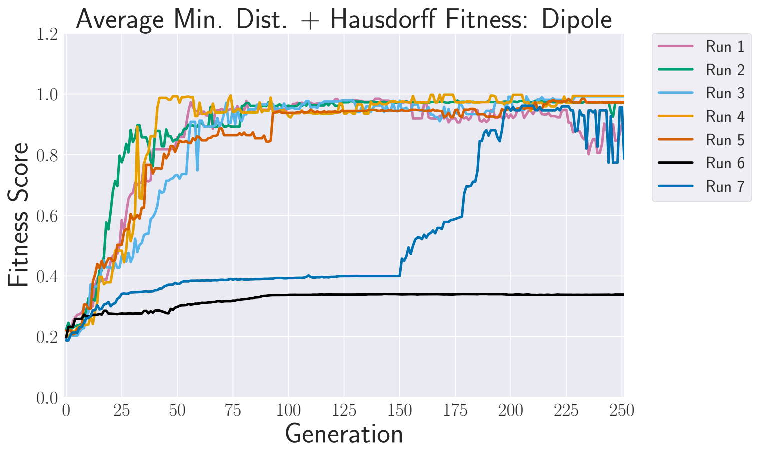


Figure 14. Results of repeated runs using the weighted sum of average minimum distance and Hausdorff distance fitness score metric for shape-to-shape evolution to the target dipole design.

The runs that rapidly achieved near-optimal solutions were those that identified primitive shape types that closely matched those of the target dipole design (cylinders or cones) in their initial generations. Once individuals made up of such primitives became common in the population of solutions, all that remained was fine-tuning toward the target shape design. There is again a degree of randomness to how quickly these matching primitives appear in the population, but it is clear from the differences between the two sets of results depicted in Figures 13 and 14 that the fitness score metric chosen also plays a role in guiding the individuals in early generations to favor the correct primitives. These results underscore the importance of further study of optimal and dynamic operator and selection method percentages to encourage efficient performance of the GA when used with each fitness function.

III. Blender Computation Time

Studies were conducted to understand the computation time incurred by the stages of the GA that rely on the use of Blender to model individuals. Outside of Blender, the remaining steps of the GA execute in under one second total, thus modeling individuals in Blender represents the most significant slowdown in terms of computation time.

The total time to model an individual in Blender, including the time required to place individual shapes, remove overlapping volume, and combine their respective meshes into one object, was recorded for individuals made up of varying numbers of shapes. This was done over 10 trials for each benchmark number of shapes, with the average computation times for the Blender steps plotted as a function of number of shapes, as seen in Figure 15. Note that this analysis was carried out locally on a CPU, which is expected to represent the fastest case.

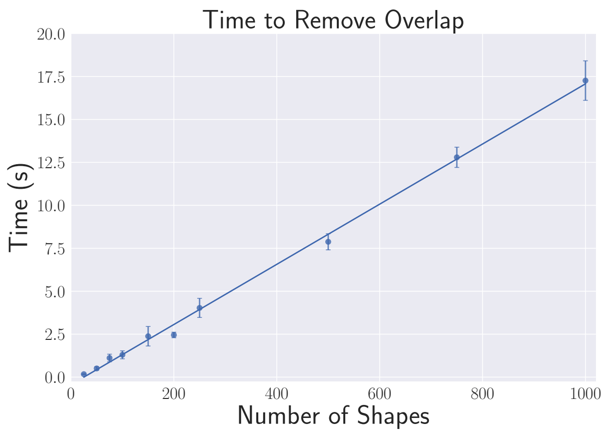


Figure 15. Average computation time of Blender modeling as a function of number of shapes in an individual when run locally.

The most significant contribution to the computation time is combining shapes into a single mesh object. A method to decrease the computation time of this process through multi-threading on multiple CPU cores was investigated. This process entailed splitting the individual being modelled into sections with an equal number of shapes, using one core per section. Each section was modelled in a separate instance of Blender prior to being recombined to form the finished model of the individual.

The results of implementing the proposed multi-threading method for a benchmark case of individuals made up of 1000 shapes are shown in Figure 16. Here, 10 trials were conducted using this method at each number of cores between 1 and 8, and the average per individual computation time over all trials was calculated and shown as a function of number of cores. This multi-threading method lessened computation time of the Blender modeling steps with increased numbers of cores used, with an approximate computation time of 14 seconds with one core decreasing to approximately 9 seconds with 8 cores.

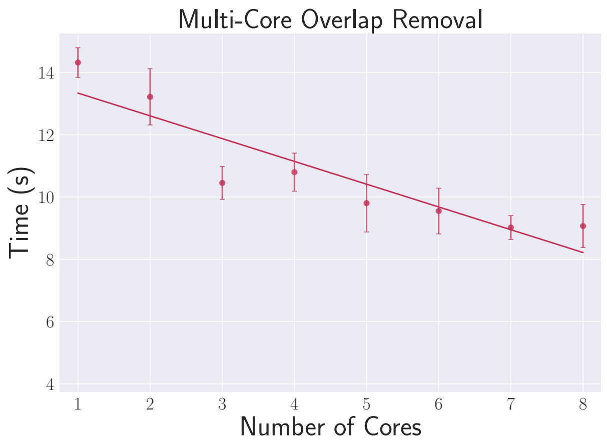
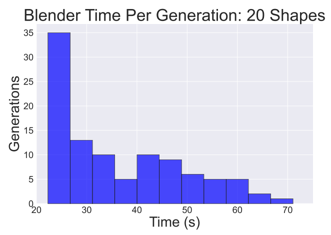


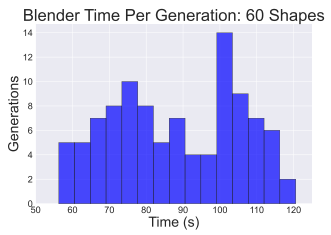
Figure 16. Average computation time of shape combination in Blender as a function of CPU cores in a multi-threaded process for individuals with 1000 shapes.

For the use cases described in this report, a parallel computing method achieved satisfactory computation times without implementing the proposed multi-threading process. The parallel computing method entailed the submission of 7 batch jobs running in parallel, each responsible for modeling up to 40 individuals.

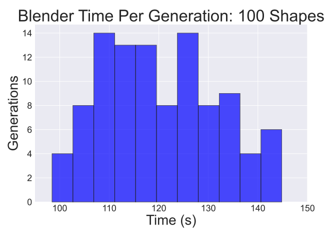
A further study was conducted to quantify the computation times of the Blender modeling of 250 individuals per generation made up of varying numbers of shapes. This is an expansion of the results shown in Figure 15, exploring total time per generation (not time for single individuals) and using a cluster instead of a local machine. The distribution of total time per generation for 100 generations shown in Figure 17 for benchmarks of individuals made up of 20, 60, and 100 shapes. The 20-shape benchmark represents the upper-limit of computation time of the modeling stages of the results presented in this report, as this was the maximum number of shapes that any individual was allowed to have, with the majority having fewer. The 60- and 100-shape benchmarks are intended to provide an expectation on the modeling speeds in Blender of other potential use cases requiring more complicated individuals. Blender modeling computation times are seen to increase with number of shapes used in the individuals of a generation; however, times remain manageable for each of the benchmark shape values when noting that the outlier values are due to wait times when queuing batch jobs.



(a)



(b)



(c)

Figure 17. Blender computation time per generation, in seconds, for the individual modeling process in Blender with individuals made up of 20 (a), 60 (b), and 100 (c) shapes. The bin width is 5 seconds for each histogram.