Two-Dimensional Topology Optimization Using Genetic Algorithms
AE505 - Final Project

Zane Morris

18/04/2022

Contents

1 Introduction

Mechanical Engineering is a faculty of applied science in which the impetus for application of optimization algorithms is driven by its unique ability to tangibly realize the result of optimization. Other faculties, exciting in their own right, frequently rely on mechanical engineers, machinists, mechanics and fabricators to realize the result of their own optimization algorithms. With this in mind, optimization within Mechanical Engineering is enabling improvement in realization of optimization in a multitude of academic faculties.

2 Topology Optimization

Structural Optimization (SO) is a popular subset of optimization problems within the field of Mechanical Engineering. SO in general, is the optimization of an attribute or property of a physical object, that satisfies a set of metrics for mechanical performance.

Topology Optimization (TO), itself a subset of SO, seeks to optimize attributes and/or mechanical performance $f(x)$ of an object, subject to attribute or performance-based constraints $g(x)$ via material existence/inexistence.

Numerical implementation of TO requires a continuum object Ω in $\mathbb{R}^p$ be discretized into an equivalent finite element (FE) model. A FE model, depicted in Figure 1 below, consists of n ($nelx$ x $nely$) finite elements and m nodes. Each node is given, by default, p degrees of freedom.

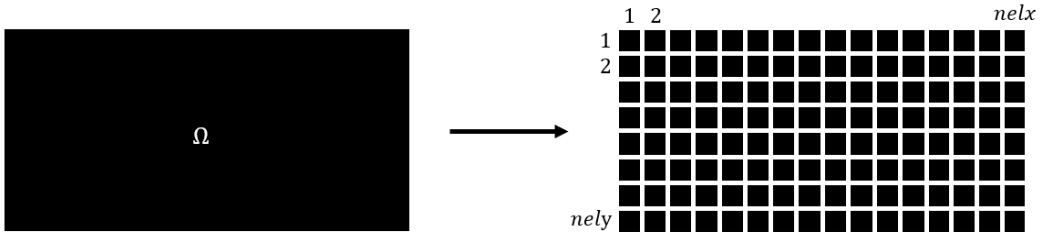


Figure 1: Discretization of Design Volume

The displacement field in the FE model is then given by the following set of vectors.

$$\begin{gathered} u_i = [u_{ix}, u_{iy}, u_{iz}]^T, \text{ where } i = 1, 2, ..., m \\ U = [u_1, u_2, ..., u_3]^T \end{gathered} \tag{1}$$

Boundary conditions and external loads are then applied to the nodes, in a manner that sufficiently constrain the model. For TO then, the set of design variables represents the degree of material existence in each finite element. A design vector $[x]_{n\text{x}1}$is then defined, housing all design variable information.

2.1 Generic TO Problem Statement

The state of a mechanical system is wholly described by the set of governing equations acting on it at any one time (i.e. Mechanics, Electromagnetics, Thermodynamics, etc.). For the purposes of basic TO, the laws of classical mechanics is considered to sufficiently govern the body, and in the case of a sufficiently constrained, linear static FE model, Newton's 2nd Law is used.

Newton's 2nd law for a constrained rigid body states that $\sum F = 0$, where F represents externally applied loads. In the case of a deformable body, one must consider the internal loads, f_i. An FE model's internal loads are extracted at each node, using Hooke's law to model deformation in an element. Each node is assumed to deform a displacement u_i, each element is assigned a stiffness matrix $k_e(x)$, and the assembly of all element stiffness

matrices forms the global stiffness matrix $K(x)$. The balance of external and internal loads thus provides the following definition for static equilibrium.

$$K(x) \cdot U = F \tag{2}$$

U is the [$2m$ x 1] nodal displacement vector, and F is now the [$2m$ x 1] external load vector. By rearranging Equation 2, the state of the deformable object U can be determined.

The physical property that is represented by the objective function $f(x)$ is commonly reliant on the state of the system, that is, $f(x, U)$. Additionally, this state is itself dependent on the design variable x, as the existence/nonexistence of material will alter the set of element stiffness matrices, therefore altering the deformation of a constrained body, thus $f = f(x, U(x))$.

The generic problem statement for a TO problem is given below, where Equation 2 is satisfied implicitly though the displacement vector's dependence on x. [1]

$$\begin{aligned} \min_{x} \quad & f(x, U(x)) \\ \text{s.t.} \quad & g_i(x, U(x)) \leq 0, \quad i = \{1, ..., p\} \\ & g_j(x, U(x)) = 0, \quad j = \{1, ..., q\} \end{aligned} \tag{3}$$

The goal of TO is to return an optimal, binary distribution of mass within the original design volume (DV). This is summarized below for the i^{th} voxel, where as x is the final solution.

$$x_i = \begin{cases} 1, & \text{for material} \\ 0, & \text{for void region} \end{cases}, \text{where } i = 1, 2, ..., n \tag{4}$$

Gradient-based algorithms are considered to be the classic approach to TO, and has been since its inception. More recently, Sigmund et. al. published an Optimality Criterion (OC) method for simple, two-dimensional (2D) DVs. [2] This later method, although limited to a regular mesh, is robust and efficient in solving basic geometry with an array of boundary conditions and constraints. While the goal of TO is to return a binary distribution of mass in the DV, both of these methods make use a continuous remapping of design variables called Solid Isotropic Material Penalization (SIMP).

3 Power Law Methods

The objective functions often used in TO are algebraically simple, and can be easily differentiated. If the feasible domain of the design space is continuous, sensitivity analysis can be conducted, and classic gradient-based methods such as Conjugate Gradient Descent Method with exterior penalties, SLP, SQP, and MMA can be utilized.

To inject continuity into the design space, material properties of the elements are considered. Each element in a single material body will have density ρ and Young's Modulus E. Being that the existence of density is synonymous with existence of material, Sigmund proposed that TO design variables x_i be representative of element density ρ_i, and that these element densities be used to define a continuous relationship with mechanical properties such as Young's Modulus (i.e. $E = E(\rho)$).

To make this work, two considerations must be made. Densities must be mapped to a half-open domain (0,1], where 1 represents default material density. This is done so as to not artificially amplify material stiffness. The physical realization of elements where $0 < x < 1$ is not possible; intermediate density values are meaningless, and invalidate any solution with a significant portion of the final body defined by them. Sigmund uses a penalty exponent p which drives intermediate solutions to either 0 or 1. The value of p is commonly taken to be 3, as this

presents a good balance between performance (highly binary final solution) and numeric stability (nonlinearity). The final form of this continuous relation is seen below.

$$E_i(x) = x_i^p E_0 \tag{5}$$

$E(x)$ is the Young's Modulus passed to the global stiffness matrix $K(x)$, and is then used to compute the displacement field, and objective functions that are dependent on system state.

Historically, TO is prone to numeric instabilities known as checkerboarding. Checkerboarding are manifestations of geometries that are infeasible to manufacture, and are characterized by spatial patterns of filled (1) and void (0) elements. A solver, void of any manufacturability constraints, may oscillate between different checkered solutions, never satisfying particular convergence criterion.

To alleviate this issue, a convolutional filter is applied to the objective sensitivity $\frac{\partial f}{\partial x}$, imparting a more continuous transition between filled and void regions. The mesh-independent filter below updates the sensitivities.

$$\begin{aligned} \frac{\hat{\partial} f}{\partial x_e} &= \frac{1}{x_e \sum_{g=1}^{N} \hat{H}_g} \sum_{g=1}^{N} \hat{H}_g x_g \frac{\partial f}{\partial x_g} \\ where \quad \hat{H}_g &= max(r - dist(e, g), 0) \end{aligned} \tag{6}$$

The filtering required to realize manufacturable and realistic solver output in a continuous domain yields solutions that are itself an interpolation or filtering of the objective function. While the solver is attempting to converge on the global optima, the filtering causes deviation from optima.

Additionally, some of the most advanced gradient-based solvers (MMA, SQP, etc.) require extensive tuning and experimentation. Even with tuning, the mathematical dependency of gradient-based solvers are still capable of premature convergence to local optima for non-convex objectives. Often times gradient-based solvers are initialized at a set of different initial conditions, so as to explore the feasible space more completely. In design hyperspaces, it is nearly impossible to discern a visible shape of the objective function, and this is one of the few ways to ensure global optimality.

As an alternative to issues surrounding power-law OC and gradient-based solvers, heuristic optimization methods such as Genetic Algorithms (GA) have been implemented as a TO solver engine. The decline of computational inefficiency in GAs for large-scale TO problems, relative to gradient-based solvers, is well documented. They do however escape the required filtering in continuous reformulations of design variables, and thus are likely, given sufficient time and variation, to converge on global optima. [3]

4 Genetic Algorithms

The family of gradient-free optimization methods, GAs being one the prominent constituents, take advantage of a paradigmatic shift away from single-point search methods used by gradient based solvers. Rather, GAs make use of a *population* of candidate solutions (individuals), which is revised iteratively by a set of heuristics.

Genetic Algorithms attempt to utilize the principles of biological evolutionary theory to iteratively improve the fitness or strength of the overall population. GAs can handle non-smooth objective functions, as gradient dependency is null. Considered to be a robust "brute force" method, GAs are capable of solving many more problems than gradient-based solvers, coming at the expensive of computation; large populations and high-dimensionality design spaces require additional memory and extended solve times. If sufficient computational resources are available, and complexity of the objective function warrants it, GAs typically discover global optima, outperforming gradient-based solvers who are prone to early convergence to local optima.

The general structure of a GA is seen below. Each subroutine is further described in a dedicated section of the document.

Algorithm 1 Genetic Algorithm

```
while Convergence Criterion (Sect. 4.6) do
    state_processing() (Sect. 4.2)
    objective() (Sect. 4.2)
    elitism() (Sect. 4.3)
    selection() (Sect. 4.3)
    crossover() (Sect. 4.4)
    mutation() (Sect. 4.5)
end while
```

Each iteration in the optimization algorithm is called a "generation", within which three operations take place in sequential order. These are:

Selection: High performing individuals are grouped into a mating pool. These members are denoted "parents".

Crossover: Each pair within the mating pool will spawn an equal number of offspring, which will temporarily replace the parent members. The offspring are denoted "children". Following reproduction, parent members are removed from the population, therefore yielding a completely new population made up of children.

Mutation: An insurance for genetic diversity, so as to avoid local optimality, children are probabilistically subject to slight variations in genetic makeup, that were not directly descendent from parent members.

4.1 Genetic Algorithms for TO (GATO)

Considering now the GA for TO, it is prudent to use a gradient-based optimization framework as a datum for both solver processing requirements, and functionality.

The continuous voxel density variables used by gradient-based SIMP methods maintain a solid connection between all elements in the DV. The half-open domain $\rho \in (0, 1]$ is used to both ensure this universal connectedness, as well as remove risk of matrix singularity when inverting K in Equation 2 to solve for U.

The bit-encoded GA cannot accomplish an equivalent feat; voxel density is defined strictly by material existence (1) and inexistence (0). Rather than encoding the real-valued density in SIMP, the encoding is a direct representation of material existence in a 2D voxelized structure. The existence on 0s in an individual inherently injects discontinuity in the FE model.

Published research into this design connectivity issue for GATO failed to propose a method that properly biases the convergence of disconnected individuals to connected ones. Rather, the methods implement analyses that increase representation degeneracy; the full breadth of an individual's genetic info (material distribution) is selectively truncated, and population diversity is reduced.

4.2 Proposed Methodology

A paper authored by Wang et. al. titled *Structural topology design optimization using Genetic Algorithms with a bit-array representation* proposes a GA with an improved set of heuristics, aiming to accomplish accelerated convergence of disconnected to connected topologies, while maintaining acceptable levels of diversity. [4] The remainder of this section will discuss the theory and logic of the method. The following sections of the report define a problem to be solved with this method, followed by its testing, analysis, and review.

Consider a 2D voxelized design space, with randomly distributed filled (1) and void (0) voxels. GATO's inherent lack of continuity between voxels necessitates that the constraint defined below be added to a problem statement.

$$n_c - 1 = 0 \tag{7}$$

n_c is the number of connected bodies found within the 2D space of an individual. The condition $n_c > 1$ in reality is invalid, as the separately connected objects would not contribute to each other's mechanical performance.

The relevance of constraints for TO requires a revision to the objective compliance function. Since gradient information is not available in the GA, directions for minimization of the objective cannot be derived using principles of calculus. Rather, a penalty function method is commonly applied for constrained GA problems. The penalty method appends a series non-negative terms to the objective function, in which the constraint equations are mathematically represented. Each constraint term is coupled with a penalty factor Γ that is used by the designer to tune importance of constraints, relatively.

$$F(x) = f(x) + \sum_i \Gamma_i \langle g_i(x) \rangle, \quad \langle \cdot \rangle = max(0, \cdot) \tag{8}$$

The non-negative nature of penalties implies that the goal in this framework is to minimize $F(x)$. A considerable drawback of this method is the time-consuming trial and error phase of tuning penalty factors.

Deb et. al. [3] proposed a replacement of the penalty method, wherein objective function evaluations are handled by a tournament operator embedded in a conditional objective function. For this implementation, the following tournament criteria is defined.

Criteria 1: Any feasible solution is preferred to any infeasible solution.

Criteria 2: Between two feasible solutions, the one having a better objective function value is preferred.

Criteria 3: Between two infeasible solutions, the one having smaller a constraint violation is preferred.

These rules are mathematically represented through the following objective function revision.

$$F(x) = \begin{cases} f(x), & \text{if } x \in \mathbb{F} \\ \tilde{f} + viol(x), & \text{otherwise} \end{cases} \tag{9}$$

In Equation 9, $\mathbb{F}$ is the feasible subspace of the 2D design space within which all constraints are satisfied, $\tilde{f}$ is the objective function value associated with the current worst performing feasible solution, and $viol(x)$ is a place holder representing a set of simple penalty functions, that are executed based on the degree of infeasibility of a solution.

Considering a population of m members, each a binary 2D array, feasibility is categorized by the following:

Condition 1 - Geometric Connectedness: Does the union of all non-void voxels create a single connected structure.

Condition 2 - Structural Connectedness: Does a geometrically connected structure or substructure connect the nodes with boundary conditions to the load location. In other words, is the structure fully constrained.

Condition 3: Does the structure meet the inequality constraints $g_i(x)$.

Condition 1 is examined using recursive blob analysis. If $n_c = 1$, the member passes Condition 1. Condition 2, examined regardless of Condition 1, tests all connected substructures (blobs) of each population by comparing the boundary condition nodes and the blob's nodal set. If a sufficient overlap between the two sets is determined (i.e. there exists sufficient boundary conditions and blob overlap to ensure the body is fully constrained), the member passes Condition 2.

The reason that Condition 2 is tested for population members that both do and do not comply with Condition 1, is to determine the unusable amount of material in a member ($\tilde{A}$). For example, if member i has three blobs, and one of these passes Condition 2, the voxels in the passing blob are deemed "usable", and do not contribute to $\tilde{A}$.

The status of the feasibility conditions for each individual determines its violation function, seen below.

$$viol(x) = \begin{cases} \Gamma_c(n_c - 1) + \Gamma_a \tilde{A}, & \text{if Cond. 1 Failed} \\ A_0^2, & \text{if Cond. 2 Failed} \\ \Gamma_n \sum_i \langle g_i(x) \rangle, & \text{if Cond. 3 Failed} \end{cases} \tag{10}$$

In Equation 10, Γ_c, Γ_a, and Γ_n are penalty multipliers, used to assign importance to specific parameters, and A_0 is the area of the default 2D design space. The default values used in [4] are summarized below.

Table 1: Default Penalty Multipliers

Γ_c	Γ_a	Γ_n
A_0	1	1

The original penalty function method uses penalty factors to scale penalties to a similar order of magnitude as the objective function. However, the original proposal from [3] eliminated the need for penalty factors, as using the comparison criteria, two individuals were never compared by the combination of objective and penalty values; comparison was either based solely on feasibility (comparison of booleans), dependent solely on the penalty values, or solely on objective values ($f(x)$).

Geometric disconnectedness is penalized by both the number of blobs, and unusable area. Structural disconnectedness is largely penalized by the constant A_0^2, the square of design space area. This is done to rapidly eliminate floating objects. As will be seen in Section 5.2, $g_i(x)$ is replaced by a volume fraction constraint; if a member is both geometrically and structurally connected, it is only marginally penalized by a normalization of this constraint to the range [0,1].

The novelty of the ideas in [4] comes from the implementation of the tournament-embedded objective function, framed to concurrently drives multi-object individuals to single body individuals, and single body individuals towards those with structural connectedness. It does so by penalizing both n_c and $\tilde{A}$, and A_0 respectively, with emphasis placed on the former.

Under close examination, the combination of Equations 9 and 10 satisfy the tournament selection criteria implicitly, rendering the implementation of a programmed tournament framework redundant. Through definition of $\tilde{f}$, tournament Criteria 1 is satisfied, as non-negative penalties enforce infeasible solutions to have larger $F(x)$ values than any feasible solution. Criteria 2 and 3 are intuitive, and make more sense in the context of a discussion on the *selection operator*.

4.3 Selection

Selection is the operator used to determine the subset of a current population deemed to be fit for mating (mating pool), itself used to generate statistically more fit offspring. A common mating pool selection technique is the Roulette Wheel Selection (RWS), wherein probabilities for selection are generated using Equation 11.

$$P_i = \frac{F_i}{\sum_{i=1}^{n} F_i} \tag{11}$$

The property $\sum_{i=1}^{m} P_i = 1$ means that each probability can be equally represented as a range $[\text{low}_i, \text{up}_i]$ such that $P_i = \text{up}_i - \text{low}_i$, and both low_i and up_i fall within the range [0,1]. This is useful for the numeric implementation of

RWS; given N random pointers $\{r_1\, r_2, ..., r_N\}$ (floating point numbers) in the range [0,1], population member i is added to the mating pool if $\text{low}_i \leq r_i < \text{up}_i$. The addition of a member to the mating pool is therefore proportional to its fitness.

The issue concerning this method however, lies in its ability to marginalize diversity in the mating pool, and thus future GA generations. First, the degree of non-linearity or discontinuity of an objective function may cause the range of $\{F_i\}$ to stretch across orders of magnitude. Equation 11 being dependent on $\{F_i\}$ would yield probability proportions that greatly favour a selected few members, over the majority.

To relieve the mating pool of some of its homogeneity, the distribution of probability for selection is remapped based on the rank of each individual's fitness; fitness is redefined in terms of rank rather than objective function value. Given the ordered population $X = [x_1, x_2, ..., x_m]$ such that $F_1 > F_2 > ... > F_m$, fitness is redefined as $\{F_i = \text{index}(X_i)\}$. Then, the execution of Equation 11 generates a more even distribution of selection potential, while maintaining order of selection probability.

The same issue prevails, however to a lesser degree, when the selection mechanism is applied. An additional flaw in this default selection operator is that probability of selection is independent of the number of selections. Therefore if x_i is represented by $P_i = 0.8$, then probabilistically, x_i will make up 80% of the mating pool. To avoid this, a Stochastic Universal Sampling (SUS) method is invoked, wherein a single random number r_0 is generated, but m evenly distributed pointers are generated using this random number, in the range [0,1]. Using RWS, m pointers associated with m random numbers, could have generated the same number and therefore selected the same population member each time. With SUS, m pointers are evenly distributed, allowing a more diverse representation of the population in the mating pool, while withholding a representation of fitness proportionality. The pointers used by SUS are generated using Equation 12 below.

$$r_i = r_0 + q_i \Delta p \tag{12}$$

where,

$$q = \mathbb{Z} \in [0, m-1]$$
$$\Delta p = 1/(m-1)$$

A mating pool of $m(1 - p_e)$ members is generated. The p_e portion of the population is automatically carried to the next generation, with all genetic information with-held. The safekeeping of p_e members is called *elitism*, and is used in tandem with convergence criterion GA methods. Usually, employment of elitist strategies improves convergence speed, and will improve the chances of locating the optimal individual.

The mating pool then generates offspring through the *crossover operator*.

4.4 Crossover

Each pair of mating pool parents generate an equal number of children. The genetic makeup of each child is generated by a combination of the parents' genetic information (alleles) in a heuristic manner. Many sets of heuristics exist, some better suited for 1D bit-encoded design variables, others suited for multi-dimensional bit-encoded variables. One dimensional crossover operators have been applied to two dimensional TO problems, however have been seen to invoke geometric bias against the generation of vertical columns and horizontal members. [5]

The Uniform Crossover Method (UCM) applies a binary mask to the voxel structure, and selects the source and destination of the crossover accordingly. The crossover of each bit is independent of the rest, nullifying geometric biasing. This crossover algorithm's pseudocode is below.

Algorithm 2 Uniform Crossover

for $p1_{val}$, $p2_{val}$, $mask_{val}$ in parent1, parent2, mask **do**
  if $p1_{val} = mask_{val}$ **then**
    add $p1_{val}$ to c1 % c1 is child1
    add $p2_{val}$ to c2 % c2 is child2
  else
    add $p2_{val}$ to c1
    add $p1_{val}$ to c2
  end if
end for

An issue with crossover is its inability to ensure monotonic improvement of solution fitness; the rules governing the sharing of genetic information are not guaranteed to generate children that improve upon parents. To safeguard against loss of important genetic information, a percentage $(1 - p_c)$ of the mating pool is selected not to mate. This percentage is unbiased, and the algorithm randomly selects indices of the mating pool.

4.5 Mutation

The algorithm's reproduction model is prone to generating offspring with a lack of diversity. Besides avoiding this by tuning the GA's population size m, and crossover probability p_c, each bit in each binary string is subject to a small, user-defined probability p_m of bit flipping. This ensures that population members with high-similarity are slightly varied, enabling exploration of the objective function in the neighborhood of the non-mutated member. Mutation also prevents permanent loss of genetic knowledge discovered throughout the evolution of the population.

4.6 Convergence Criterion

GAs being void of gradient information, are unable to mathematically verify if an individual is at a global, local, or intermediate solution. The replacement of gradient-dependent optimality conditions is simple, intuitive metrics. The first is maximum specified iteration, and the second is a limit on the number of consecutive iterations where the elite solution does not change. If the solver terminates via case two, it is implied that the objective function within the neighborhood of the elite member has been sufficiently probed for improvement, without achieving any. It is therefore prudent to terminate the solver, as further search is expected to not return pertinent information.

5 Problem Definition

5.1 System Model

In the interest of both simplicity and consistency with academic benchmarks for solver performance, a 2D cantilevered beam has been selected as the subject of study. See the system diagram below.

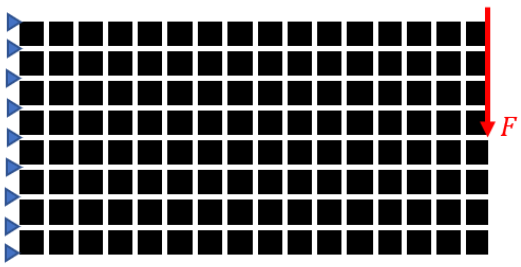


Figure 2: Cantilever Beam Loading and Boundary Conditions

The design volume (DV) is a planar surface with an aspect ratio of 2:1. The FE model is voxelized with 200 (10 x 20) regular elements.

The majority of the elements are free to translate in the x and y axes, and free to rotate about the z axis. Nodes at the leftmost edge are fixed in x and y. A vertically downward unit load is applied to the half-height of the rightmost edge of the model.

5.2 Problem Statement

The goal of the project is to minimize compliance of the cantilevered beam, such that a desired volume fraction f is achieved.

$$\begin{aligned} \min_{x} \quad & F(x) = \begin{cases} c(x, U(x)) = \sum_{e=1}^{n} u_e^T k_e(x) u_e, & \text{if } x \in \mathbb{F} \\ \tilde{c} + viol(x), & \text{otherwise} \end{cases} \\ \text{s.t.} \quad & \frac{V(x)}{V_0} = f \\ & K(x) \cdot U = F \end{aligned} \tag{13}$$

In Equation 13, $V(x)$ is the volume of the current solution, and V_0 is the DV volume.

For the remainder of the report, each condition discussed in Section 4.2, in the order they were introduced, will be referred to as c-feasible, s-feasible, and f-feasible, if the condition is met. Otherwise, the suffix on this naming convention will be replaced with "infeasible".

5.3 $\tilde{c}_0$ Determination

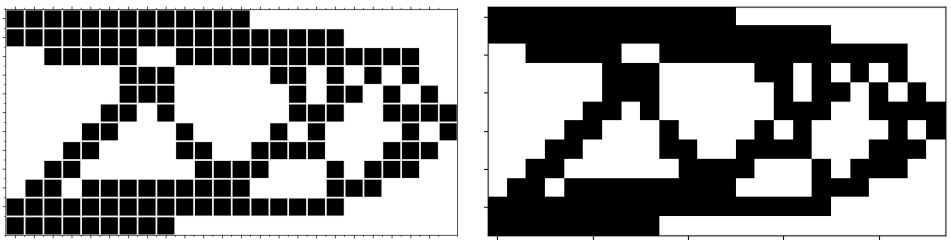
The calculation of compliance is mesh dependent, meaning FE model parameters have an indirect effect on results. For the 10 x 20 mesh size selected, an initial $\tilde{c}$ value is determined by testing an intuitively poor distribution of mass at a low desired mass fraction (0.5), and scaling by a safety factor. Besides aiding in post-processing, determining a non-arbitrarily high $\tilde{c}_0$ also improves ability to identify numerical issues if they arise.

Figure 3: Intuitive, low performance model used to estimate $\tilde{c}_0$

For the distribution of mass seen in Figure 3, compliance of 3460.736 was calculated. Given that the geometry here is an estimated worst distribution, a conservative estimate for $\tilde{c}_0$ of 5000 was used.

5.4 Solver Validation

Prior to solving the problem using randomized or unintelligent distributions of mass, the functionality of the Python interpretation of the methods proposed in [4] was validated. Using the authors' documented optimal solution as a starting condition, the algorithm was executed and results are below.



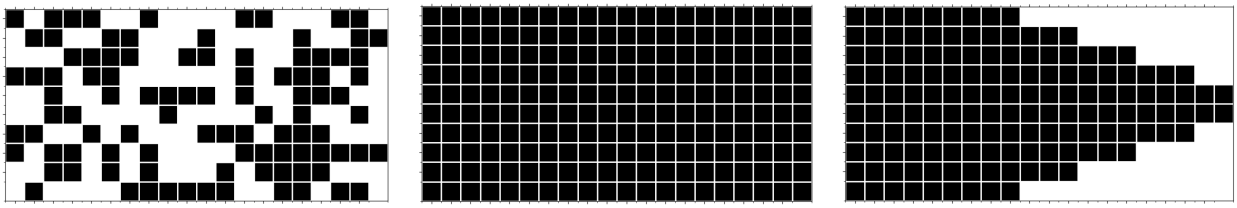
(a) Solution in [4]: $c(x) = 69.39$ (b) Python solution: $c(x) = 69.387$

Figure 4: Comparison of Results in [4] and Interpreted Methods

This test was conducted with $i_{max} = 1000$, $p_e = 0.1$, $p_c = 0.9$, $p_m = 0.01$, and $f = 0.5$. The 12 x 24 voxel structures show no difference in final geometry, and yield similar compliance values. Given an initially feasible solution then, the Python interpretation works as expected.

5.5 Problem Setup

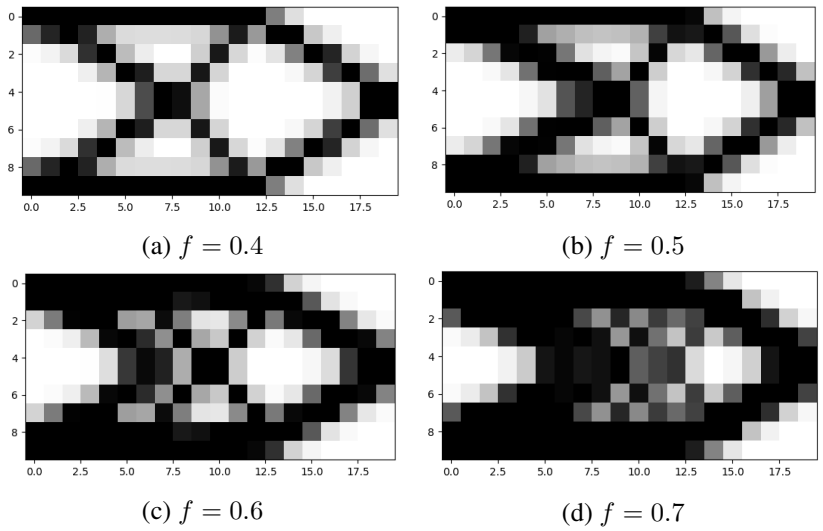
The subsequent analysis will consist of simulations with three initial conditions, hereafter referred to as *random*, *solid*, and *triangle*. These are summarized below.



(a) *random* (b) *solid* (c) *triangle*

Figure 5: Initial Conditions for Cantilever Beam

A baseline for performance is defined using the 99 Line TO code. Topological results for four volume fractions can be seen in Figure 6, and quantitative performance summarized in Table 2. To generate these results, a filter radius of 1.1, and a SIMP penalty value of 3 were used. [2]



(a) $f = 0.4$ (b) $f = 0.5$

(c) $f = 0.6$ (d) $f = 0.7$

Figure 6: 99 Line TO Baseline Geometric Results

Table 2: 99 Line Baseline Computational Performance

f	$c(x)$	Iters. & Fnc. Evals
0.4	110.313	34
0.5	73.55	26
0.6	58.852	54
0.7	50.231	42

6 Results

The entire set of GA tests conducted can be found tabulated in Appendix A. The optimal results pertaining to each of the tested volume fractions with each of the starting conditions will be discussed in this section, followed by comparison between them. Transient analysis of the solver will be explored in Section 7.

The GA parameters used in [4], tabulated in Table 3, were selected as default values.

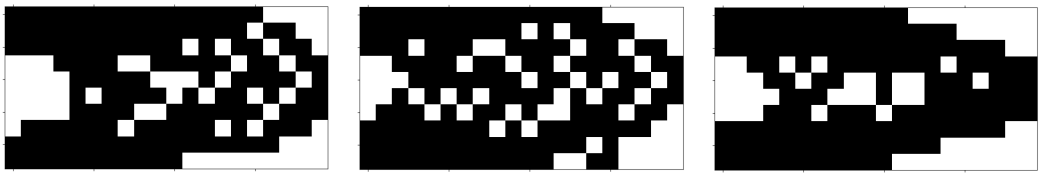
Table 3: GA parameters as per [4]

m	i_{max}	p_e	p_c	p_m
100	1000	0.1	0.9	0.01

Revision of these parameters was completed as seen fit. Through trial and error, it was determined that p_e and p_m be changed to 0.05 and 0.001 respectively. The reasoning for these revisions is discussed in more detail in Section 7.

6.1 Volume Fraction of 0.7

The optimal 2D cantilever beam topologies of $f = 0.7$, with the associated compliance computations are below.



(a) *Random* initialization $c(x) = 48.267$

(b) *Solid* initialization $c(x) = 48.681$

(c) *Triangle* initialization $c(x) = 49.216$

Figure 7: Optimal Topologies for $f = 0.7$

With minimal review, it is apparent that common geometric features exist in the solutions. Of these similarities, perhaps the easiest to identify is the removal of material at the two right-most corners of the space. This region is not located along any intuitive load path between external loads and constraints. By virtue of this attribute, its existence adds marginal mechanical advantage, and can be unanimously removed. The second common feature is the void space at the centre of the left border. Given the symmetry of the design space and load case, bending stress will be induced about the horizontal centroidal axes. The classical mechanics model of bending stress suggests that stress will linearly decay towards the centroid, where it becomes 0. This approximately symmetric void region is where bending stress is minimal, making it a viable region to eliminate mass. These features are consistent with the baseline geometry in Figure 6, however the remaining areas where the GA removed mass are inconsistent; large amounts of checkerboarding manifest in the *random* and *solid* instantiations, which present both numeric inconsistencies and manufacturing infeasibilities. The FEA in these cases rely on node-node connections

between elements that are not true neighbours. This type of structure is not easily realizable for most engineering applications.

The unique features, non-realizable as they may be, highlight the multi-modality of the space; significantly different design vectors yield similar objective values, indicative of multiple local optima.

The GA highlights its ability to outperform the 99 Line TO solution, decreasing compliance by 3.91%, 3.08%, and 2.02%. These improvements however, come at the cost of computation efficiency. For the sake of concision, convergence trend plots have been included in Appendix B of the report. Table 4 summarizes the computational performance of each test. For these tests, $i_{max} = 1500$.

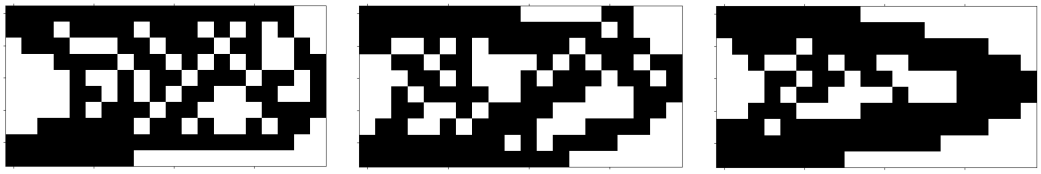
Table 4: GA computational performance with $f = 0.7$

init cond.	iters.	$F(x)$ executions	$\bar{t}$ per Gen.
random	810	81000	3.5
solid	1500	150000	3.8
triangle	1000	100000	3.78

It is clear that the biologically-inspired heuristics require far greater computational resources, to return a result than marginally outperforms power-law OC and gradient-based solvers.

6.2 Volume Fraction of 0.6

The optimal 2D cantilever beam topologies of $f = 0.6$, with the associated compliance computations are below.



(a) *Random* initialization $c(x) = 60.136$

(b) *Solid* initialization $c(x) = 59.323$

(c) *Triangle* initialization $c(x) = 60.27$

Figure 8: Optimal Topologies for $f = 0.6$

Similar trends to that of the $f = 0.7$ solutions are seen; removal of material at the right boundary corners and left boundary's center are the first locations to become void. Clear vertical asymmetry is seen in the top and bottom rows of the design space. Figure 8a shows more material at the top, while Figure 8b conversely uses more material at the bottom. The symmetry of the loading conditions suggest that this should not be the case, and the emergence of this asymmetry is hypothesized to be caused by the bias towards high-fitness individuals early in simulation. The universal crossover operator evaluates bit flipping at each voxel individually, and thus is unlikely to revert the solution back to a symmetric one. The exception to this asymmetry is Figure 8c. In this case, the solver was instantiated with a geometry having a very similar periphery to that of the solution. The solver seems to limit its search to the space defined within the border.

With 40% of the original mass now removed, diagonal truss-like members are beginning to appear in the interior region of the design space. The existence of these features are validated by their existence in Figure 6. In the GA solutions however, these members are plagued by checkerboarding. A mesh-independent filter, the likes of Equation 6 can also be applied to GAs, however a Heaviside filter/mapping must be applied to retain the binary nature of the design variables.

At $f = 0.6$, the GA fails to improve upon the baseline power-law approach. Although very similar in terms of

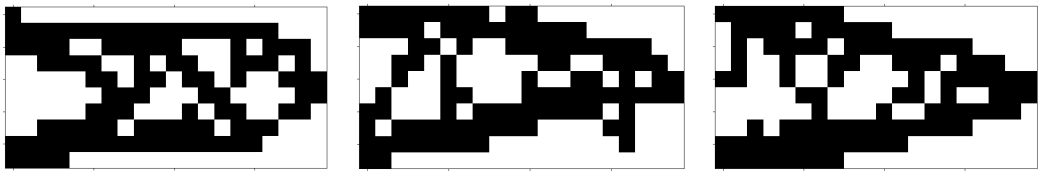
compliance (2.18%, 0.8%, and 2.41% increases), the computational inefficiency as verified by Table 5, as well as the lack of solution improvement, does not warrant the use of the GA. For these tests, $i_{max} = 1200$.

Table 5: GA computational performance with $f = 0.6$

init cond.	iters.	$F(x)executions$	$\bar{t}$ per Gen.
random	729	72900	3.61
solid	1073	107300	3.74
tri	1000	100000	3.76

6.3 Volume Fraction of 0.5

The optimal 2D cantilever beam topologies of $f = 0.5$, with the associated compliance computations are below. For problems with $f \leq 0.5$, the population size was changed from the default value, to either 120 or 130, simply to increase diversity and search potential.



(a) *Random* initialization $c(x) = 84.699$

(b) *Solid* initialization $c(x) = 93.18$

(c) *Triangle* initialization $c(x) = 80.03$

Figure 9: Optimal Topologies for $f = 0.5$

Similar features arise for $f = 0.5$ as they have for $f = 0.6$ and $f = 0.7$. More prominent now are homogeneously void regions within the interior of the final geometry. The larger percentage of material reduction has directly decreased the amount of checkerboarding possible in the solution.

Some vertical asymmetry still exists at the periphery of the design, however now less prominent.

Most prominent in Figure 9a are the diagonal members connecting the top and bottom region geometry. While the relatively coarse resolution of the design space, imposed to a degree by the computational inefficiency of the GA for this large-scale problem, may hinder the clear diagonal member geometry, the bigger culprit may be the structure of the GA itself. The focus on population diversity has, decisively so, reduced the biasing the GA needs to converge on a cleaner solution.

The bottom right corner of Figure 9b shows a short member hanging beneath the majority of the structure. This material is being under-utilized where it sits, and intuitively should be redistributed to a location inside the structure. Although it is difficult to observe the degree of diversity in a population with the size of 100, it would likely take the GA multiple generations to achieve this re-purposing. The universal crossover method employed here will swap one bit at a time, and not a set of bits, meaning that this under-utilized feature cannot be directly translated to a new location via crossover. This shortcoming highlights the requirement for large diversity and population size in a GA, as well as the intrinsic lack of intelligence embodied by the simple heuristics, compared to gradient-based approaches.

The compliance values associated with the three solutions above are larger than the baseline's $c(x) = 73.55$ solution by 15.52%, 26.69%, and 8.81% , respectively.

Repeating the solver with either the same or slightly different parameters is likely to improve the performance; time and computing resource limitations in this project limited repeated runs of the same GA to 3. For contrast, Wang ran 20 independent GA simulations with the same parameters to attain the result presented in Figure 4a. [4]

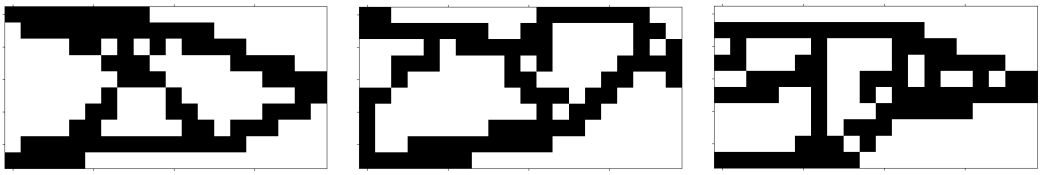
Summarizing the computational performance of the solver at $f = 0.5$ is Table 6 below. It should be noted that average generation times $\bar{t}$ were influenced by an increase in population size for this volume fraction, as it was expected that greater diversity would aid the solver in attaining initial feasibility, giving it more time to search the feasible space, unencumbered by penalties.

Table 6: GA computational performance with $f = 0.5$

init cond.	iters.	$F(x)$ executions	$\bar{t}$ per Gen.
random	908	90800	4.03
solid	998	99800	4.2
tri	545	54500	3.76

6.4 Volume Fraction of 0.4

The optimal 2D cantilever beam topologies of $f = 0.4$, with the associated compliance computations are below.



(a) *Random* initialization $c(x) = 94.639$

(b) *Solid* initialization $c(x) = 135.485$

(c) *Triangle* initialization $c(x) = 153.726$

Figure 10: Optimal Topologies for $f = 0.4$

Figure 10a shares the most features with the $f = 0.4$ solution in Figure 6; triangular substructures, large symmetric void regions, and connection at the two corners of the constrained region. Figure 10b has multi-axis asymmetry, and Figure 10c includes vertical and horizontal interior members, which although orthogonal with one another, are more prone to shear in the current load configuration.

Relative to the 99 line TO solution in Table 2, the GA achieved a 16.56% reduction in compliance with the *random* starting conditions, which is also shows highly-manufacturable geometry, but returned 22.82% and 39.354% increases with *solid* and *triangle* conditions, respectively.

Table 7: GA computational performance with $f = 0.4$

init cond.	iters.	$F(x)$ executions	$\bar{t}$ per Gen.
random	2000	200000	4.81
solid	1796	179600	5.02
tri	797	79700	5.1

The average generation solve time specified by Wang for the default parameters shown in Table 3 is 3.9s. [4]. The implementation of the algorithms and logic outline in [4] are on par with this value. The 65% increase in this project's PC clock speed suggests that the structure of the code is not as efficient as that used in [4]. Looking forward, utilization of more professional, streamlined Python modules may improve the speed.

6.5 Summary

The GA's regression from the 99 Line TO baseline is summarized graphically in Figure 11.

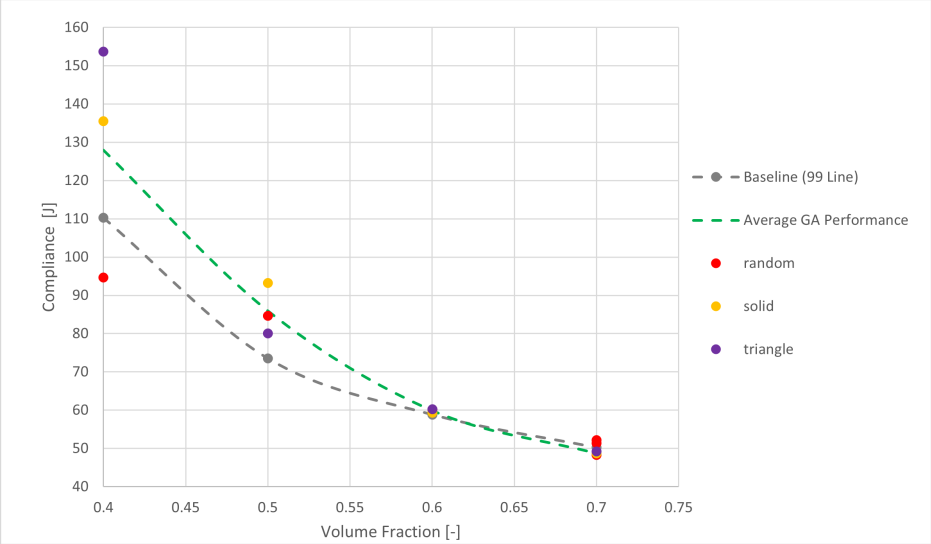


Figure 11: Performance of GA relative to 99 Line TO solutions, with respect to each starting condition

The mean of GA performance is taken for the set of tests at each volume fraction, and plotted in the green dashed line. With an overall improvement upon the OC methods for $f = 0.7$, its trend indicates a steadily declining performance (higher average compliance) as f is reduced. As mentioned before, the small sample size may not be completely representative of the true performance of the algorithm, due to inherent randomness. Future work may verify this claim through much more testing. The existence of solutions that outperform the OC method at both $f = 0.4$ and $f = 0.7$ suggest that the GA is truly capable of returning desired improvements in compliance. Wang et. al. did not specify the percentage of GA executions failing to achieve performance improvements. With this, it can be said that the GA implemented here behaves and performs as expected, with varying degrees of intuitive design and manufacturability.

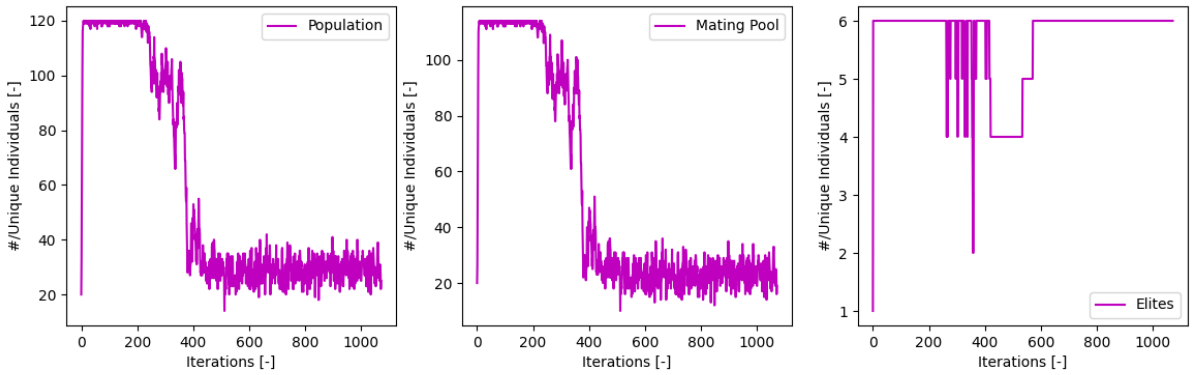
Although it was expected that the GA would outperform the OC methods more frequently than seen, the difficulty in doing so was expected to increase as f was reduced. As the amount of material in the structure decreases, so does the size of the feasible region $\mathbb{F}$. With this, the multi-modality of the solution space decreases, meaning the GA has fewer discrete optimum wells with similarly valued compliance solutions to converge on. While the plot indicates that a larger range of solutions is possible with a small volume fraction, it is expected that this increased range is only indicative of fewer existing feasible wells with similar compliance values. To this end, an improvement in performance may be made if the blob analysis could return void blobs on the periphery of the space, and eliminate a portion of them from being used in the crossover operator. This tool would focus the search for optimality to the interior of the component, ridding solutions of the requirement to consider the mechanical advantage so clearly not available at the periphery (right-most corners) of the space.

7 Discussion

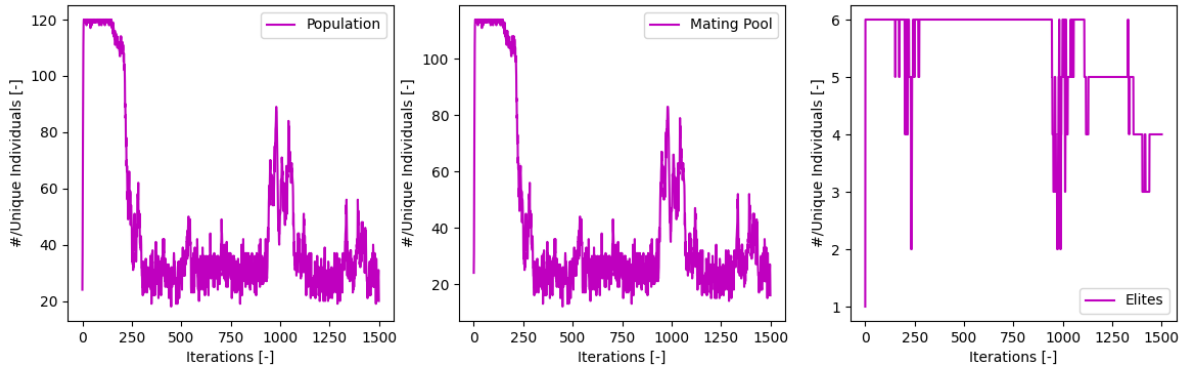
7.1 Diversity Analysis

The issue of representation degeneracy was of primary importance to the authors of [4] when building the GA framework. The genetic information of individuals with disconnected geometries was not removed, but rather penalized so that a larger portion of the space could be explored prior to a convergence. Doing this requires large diversity or uniqueness in the population.

Figure 12 describes the number of unique individuals in each the population, mating pool, and elite group, with respect to time. This transient analysis provides useful insight into the GA functionality.



(a) *Solid* initial population, with $f = 0.6$

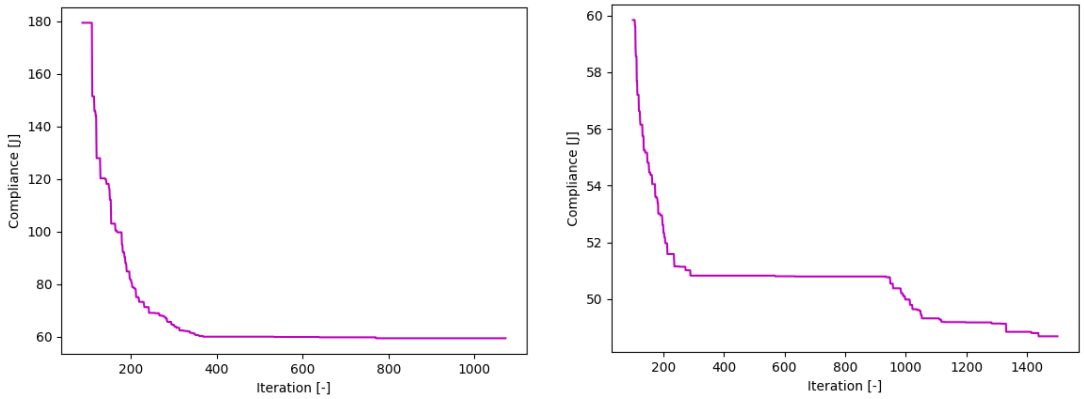


(b) *Solid* initial population, with $f = 0.7$

Figure 12: Transient trends of diversity in population (left), mating pool (center), elite group (right) for two volume fractions

In both Figure 12a and 12b, uniqueness immediately spikes to equate with the size of the population. This happens predominately through mutation, as all members in the first generation, being solid, share the majority of genetics, rendering crossover ineffectual.

In Figure 12b, a sudden drop in uniqueness of all three categories occurs at approximately $i = 250$. The uniqueness of the elite group immediately recovers, while the population and mating pool do not. The reasoning for this phenomena is best explained with the convergence plot for each of these cases, shown in Figure 13.



(a) *Solid* initial population, with $f = 0.6$

(b) *Solid* initial population, with $f = 0.7$

Figure 13: Convergence Plots from the 100^{th} iteration onward

Diversity of the population is largely responsible for determining the rate of convergence. The spikes in Figure 12b correlate to periods of improvement in the elite solution's compliance. The magnitude of the spike also directly correlates to the slope of convergence, or the rate at which compliance is improving. Convergence plateaus emerge when diversity is at its lowest. Figure 13a does not experience a similar epoch, but rather a more consistent decline of compliance.

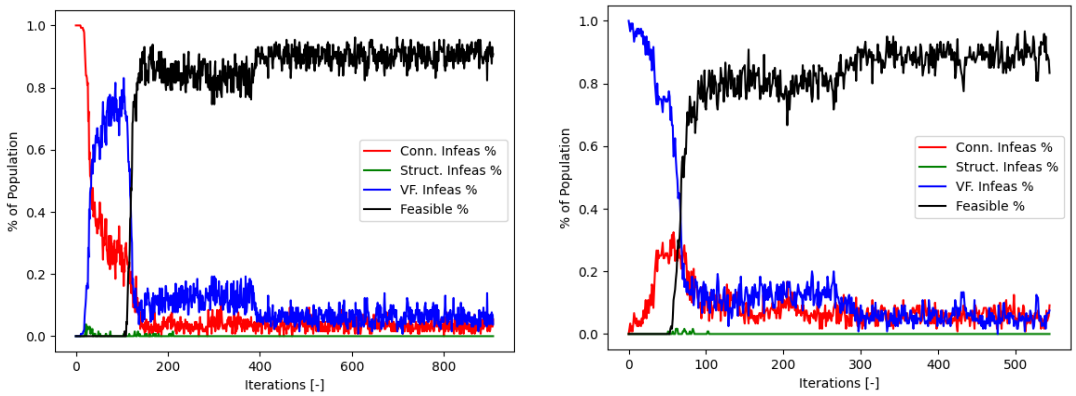
Due to the rank-based fitness mapping, and the implementation of SUS, the diversity of the population cannot drop beneath the lower band seen in the diversity trend. A selective pressure of 2 was used for the rank-based fitness mapping, meaning that the optimal solution is twice as likely to be added to the mating pool than the individual ranked in the middle.

The cyclic stages of development and dormancy seen in Figure 12b's population is mirrored by biological and natural systems. The randomness in the system makes it difficult to predict when improvement may arise, however statistically they are bound to happen. The second convergence criterion used in this model limits the length of a dormancy stage in development. A small percentage of the tests conducted could have benefited from an increase in this cap, and further exploration of its tuning is suggested.

The noise seen in the trends above is expected, due to the randomness inherent to the GA. The magnitude of noise increases following the initial decline in diversity, as the interplay between the elite group and mating pool occurs less frequently, meaning there is increased potential for replication with a duplicate individual. Noise is higher in secondary stages of development, due to increased interplay between the groups. This trait is consistent with trends from the remainder of the tests.

7.2 Serial Trend of Feasibility

The conditional penalization function used in the tested methodology often discretizes the solution sequence into distinct phases. See the two independent trends below for a *random* and *solid* starting condition.



(a) *random* initialization

(b) *solid* or *triangle* initialization

Figure 14: Categorical feasibility trend comparison

A random distribution of mass within the design volume is highly unlikely to yield a single connected structure. At the first generation, 100% of the population is considered to be infeasible due to disconnectedness, and is penalized accordingly. The disconnectedness penalty, being scaled much more than the other penalties, quickly eliminates these highly disconnected solutions, allowing the individuals with only a few disconnected objects to join via crossover and mutation. Given that the constrained voxels make up 5% of the total voxel count, it is expected that a single voxel be included in these connected objects as they show up, thereby stating that the state of structural infeasibility will be surpassed by the majority on connected solutions. This means that for the most part, a direct substitution can be made between a disconnected structure and a f-infeasible solution. This feature is shown in the first stage of Figure 14a, where the blue line is for all intents and purposes, an inversion of the red line. Once an f-infeasible solution sheds enough mass via crossover and mutation, it associates itself with a much smaller compliance than the rest of the population, meaning that rapid duplication and operation on this solution follows. The point at which one solution becomes feasible marks the end of the first phase. The second phase is where the the feasible subspace of the design space is searched.

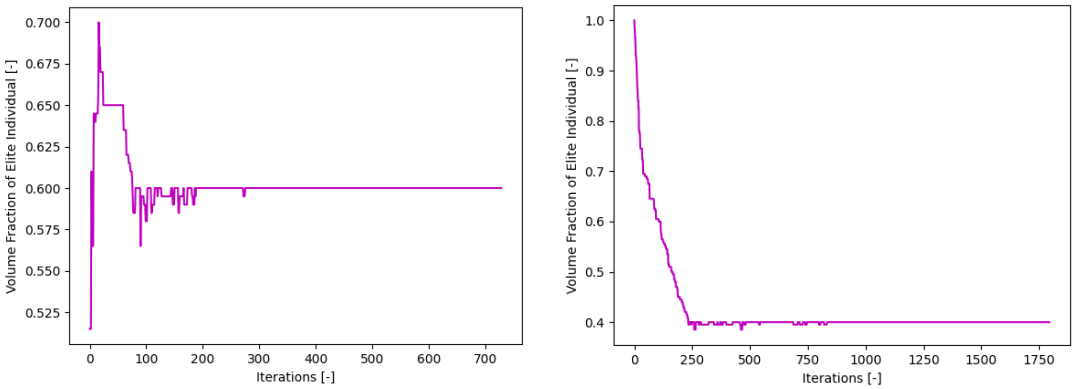
S-infeasible solutions are much more rare than any other type of infeasible solution, as $\tilde{A}$ is already penalized via Condition 1. S-feasibility, as per Equation 10 is only evaluated following confirmation of c-feasibility. This is verified in Figure 14a where non-zero portions of the population that are s-infeasible only occur after there exists c-feasible solutions.

In the case of a *solid* or *triangle* initialization, where the entire population is s-feasible, mutation drives f-infeasible solutions to c-infeasible solutions to attempt to satisfy the volume fraction constraint. The gradual diversion from a solid block, via mutation and eventually crossover, will ultimately achieve the correct volume fraction f. Following the emergence of a feasible solution, more rapidly follow, and s-infeasible solutions return to feasible ones, as material is re-added to the boundary regions. Figure 14b is an example of this case, where f-infeasible individuals are replaced by s-infeasible solutions in the first phase, followed by those being replaced by feasible solutions.

7.3 Volume Fraction Convergence Trends

Compliance is minimized when displacement is minimized. A structure with less material than another is likely to deflect more, if the material is not intelligently positioned. Intuitively then, the best TO solutions should appear with the condition $\frac{V(x)}{V_0} \simeq f$.

This property has been proven true for the entire fleet of tests conducted. The convergence on this final volume fraction is of importance for understanding how constraints are dealt with in a penalty factor format. See the two volume fraction convergence trends below, for separate tests.



(a) *random* initial population, with $f = 0.6$

(b) *solid* initial population, with $f = 0.4$

Figure 15: Volume fraction convergence trends for the elite solution

Figure 15a shows that the GA needed to increase its volume substantially in order to attain s-feasibility. Once the elite solution became completely feasible, $V(x)$ never increased above 0.6. The GA with elitism does not allow backtracking to infeasible solutions, where there may exist a better overall distribution of mass. From $100 \leq iter \leq 300$, the solver discovered optimal solutions with less than 60% volume, but consistently returned to 60%, where the stiffer, less compliant, feasible structures exist. Large solid initial populations, exemplified by Figure 15b, already having s-feasibility, purely decreases $V(x)$ until the constraint is satisfied.

Contrary to interior penalty methods for constrained optimization, the GA clearly allows for infeasible solutions, and searches for a feasible one, but cannot guarantee it. The importance of satisfying the constraints in any optimization problem should steer the designer towards either interior or exterior penalty formulations. If constraint satisfaction is vital, invoke interior penalty methods, otherwise exterior methods such as those explored here can be used.

7.4 p_e Manipulation

Values of 0.1 and 0.05 for p_e were used interchangeably throughout the course of testing. Specifically, 0.05 was used in cases where exploration of the design space was emphasized, and 0.1 was used when faster convergence was desired.

Without elitism (i.e. $p_e = 0.0$), the current optimal solution can be saved externally, however the member does not remain in the population, and has no genetically fruitful information to provide to the offspring.

7.5 Consideration for Volume Fraction as a Feasible Penalty

In early GA testing, problem executions with a small desired volume fraction (i.e. $f = 0.4$), would not break the initial feasibility threshold, $\tilde{c}_0$. Violations would steadily converge towards zero, but never become less than $\tilde{f}_0$. Initially, random populations are highly likely to be both c-infeasible and s-infeasible. The violation values for these individuals, given by Equation 10, are directly proportional to area/material used. The solver then, associating these disconnected solutions with high penalty values, is likely to remove them from the population. On the other hand, the fittest solutions in this case, are the individuals who are structurally connected, but do not meet the desired volume fraction. To meet this constraint, one must shed mass, which will likely create disconnection in the body, which then causes an increase in violation. The cyclic nature of constraint satisfaction in this problem is heavily

reliant on chance, thus requiring large populations and rich diversity for searching a large portion of the design space. Alternatively, an attempt was made to consider the violation associated with the volume fraction constraint as a "feasible" penalty, so as to enable passing of the feasibility barrier.

If the volume fraction constraint is not deemed a condition that determines feasibility, then in the current framework, any structurally connected solution is the first to become "feasible", and redefine $\tilde{c}$. If this first feasible solution happens to have a $\frac{V(x)}{V_0} > f$, and if the difference between $F(x)$ and a known optimal solution at f is larger than the volume fraction violation value, then the associated compliance will be smaller than any feasible solution at the desired volume fraction. In this case, given the GA's elitist methodology, the volume fraction constraint will never be satisfied, as the compliance must increase to satisfy the constraint. This could be corrected by increasing Γ_n, however this would be problem dependent. Therefore, a solution not conforming to the volume fraction constraint should never be deemed feasible.

Breaking the feasibility barrier at lower values of f may be solved by the reduction of mutation. Statistically, in a design space of 200 variables such as this, a $p_m = 0.01$ will flip 2 bits per individual, meaning that no individual returned by the crossover operator will be added to the population as expected. This may lead to convergence issues, where improvements made by the crossover operator are reversed/nullified by mutation. The solutions shown in Section 6 use a value of $p_m = 0.001$, whereas earlier tests used $p_m = 0.01$. With this revised value, topology solutions became more consistent and realizable.

A more involved improvement to the cyclic constraint violation issue would be to implement dynamic learning. Iteratively identify what regions in elite individuals are void, and slowly impose these on non-elites. This approach may invoke representation degeneracy, which was to be marginalized by the studied methods. However, if rapid convergence is desired, this revision may be effective.

8 Conclusions

Presented in this report was the implementation of a Genetic Algorithm used for Topology Optimization, with an improved set of heuristics, aiming to accomplish accelerated convergence of disconnected to connected topologies. The model, once validated, was applied to a 2D cantilever beam model, with which the compliance was to be optimized, subject to four different desired volume fractions. Results show improvements in compliance relative to the 99 Line OC method for high volume fractions, but increasingly poorer performance as the volume fraction is decreased. Results at these low volume fractions are more manufacturable than 99 Line solutions, due to the former's binary nature, and the existence of intermediate densities in the latter. However, the GA solutions also require multiple orders of magnitude more function evaluations, and additional system processing just to return solutions of marginally improved compliance. Gradient-based and OC optimization methods using the power-law approach are still considered to be the most rewarding solvers for this application. It is hypothesized that future correlation of GA results with 99 Line results will show improved GA performance, as a larger sample size will better represent the true capabilities of the solver.

References

[1] P. Christensen and A. Klarbring, *An Introduction to Structural Optimization*. Solid Mechanics and Its Applications, Springer Netherlands, 2008.

[2] O. Sigmund, "Sigmund, o.: A 99 line topology optimization code written in matlab. structural and multidisciplinary optimization 21, 120-127," *Structural and Multidisciplinary Optimization*, vol. 21, pp. 120–127, 04 2001.

[3] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2, pp. 311–338, 2000.

[4] S. Wang and K. Tai, "Structural topology design optimization using genetic algorithms with a bit-array representation," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 36, pp. 3749–3770, 2005.

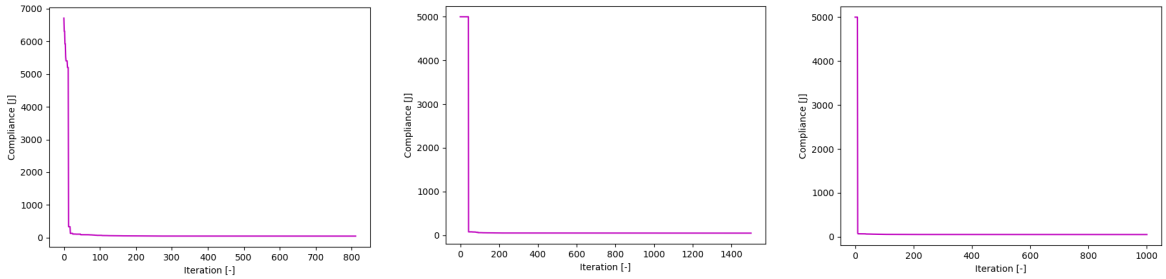
[5] G. Winter, *Genetic algorithms in engineering and Computer Science*. John Wiley amp; Sons, 1995.

9 Appendix A: Tabulated Test Results

Table 8: 2D TO GA Results

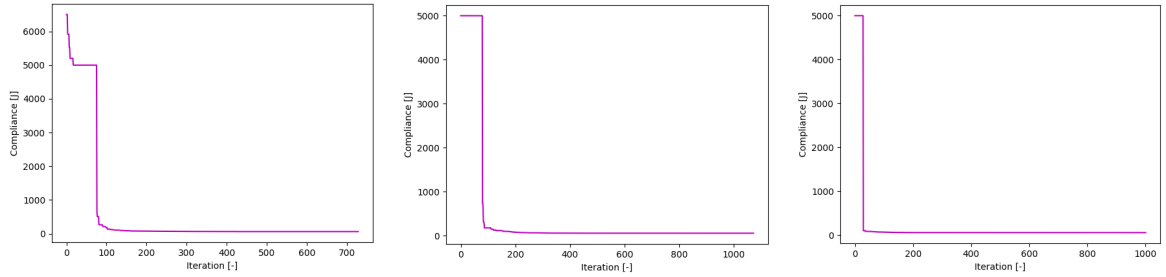
	Parameters						Performance		
Run ID	Init. Pop.	Pop. Size	p_e	p_c	p_e	f	$c(x)$	iters	fnc. evals
5	tri	50	0.1	0.9	0.01	0.7	59.104	100	5000
6	99opt	50	0.1	0.9	0.01	0.7	60.567	100	5000
7	paper-opt	50	0.1	0.9	0.01	0.7	68.278	100	5000
8	paper-opt	50	0.1	0.9	0.01	0.7	69.387	100	5000
9	solid	50	0.1	0.9	0.01	0.7	77.765	100	5000
12	tri	50	0.1	0.9	0.01	0.6	59.104	323	32300
13	random	75	0.05	0.9	0.01	0.7	64.483	630	63000
14	random	75	0.05	0.9	0.01	0.7	65.111	980	73500
15	random	75	0.05	0.9	0.01	0.7	69.438	1000	75000
16	random	75	0.05	0.9	0.01	0.5	70.42	1000	75000
17	random	75	0.05	0.9	0.01	0.7	69.712	1000	75000
30	random	120	0.1	0.9	0.001	0.7	52.133	1000	120000
31	random	120	0.1	0.9	0.001	0.7	51.402	1000	120000
32	opt from paper	100	0.1	0.9	0.001	0.5	68.983	991	99100
33	random	120	0.1	0.9	0.001	0.7	52.133	1000	120000
34	random	120	0.05	0.9	0.001	0.7	48.267	810	97200
35	random	120	0.05	0.9	0.001	0.6	60.136	729	87480
36	random	130	0.05	0.9	0.001	0.5	84.699	908	118040
37	random	130	0.1	0.9	0.001	0.5	95.23	878	114140
38	random	130	0.05	0.9	0.001	0.6	69.288	906	117780
39	solid	120	0.05	0.9	0.001	0.7	48.681	1500	180000
40	solid	120	0.05	0.9	0.001	0.6	59.323	1073	128760
41	solid	120	0.05	0.9	0.001	0.5	93.18	998	119760
42	tri	120	0.05	0.9	0.001	0.7	49.216	1000	120000
43	tri	120	0.05	0.9	0.001	0.6	60.27	1000	120000
44	tri	120	0.05	0.9	0.001	0.5	80.03	545	65400
45	tri	140	0.05	0.9	0.001	0.4	153.726	797	111580
46	random	140	0.05	0.9	0.003	0.4	94.639	2000	280000
47	solid	140	0.05	0.9	0.003	0.4	135.485	1796	251440

10 Appendix B: Convergence Plots



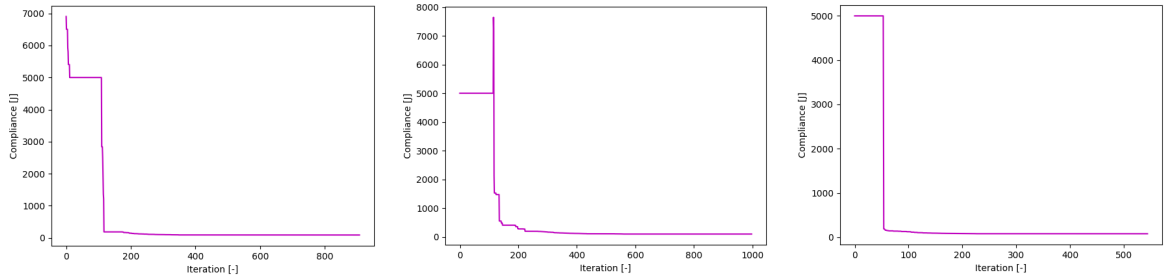
(a) *random* (b) *solid* (c) *triangle*

Figure 16: Convergence Trends for $f = 0.7$



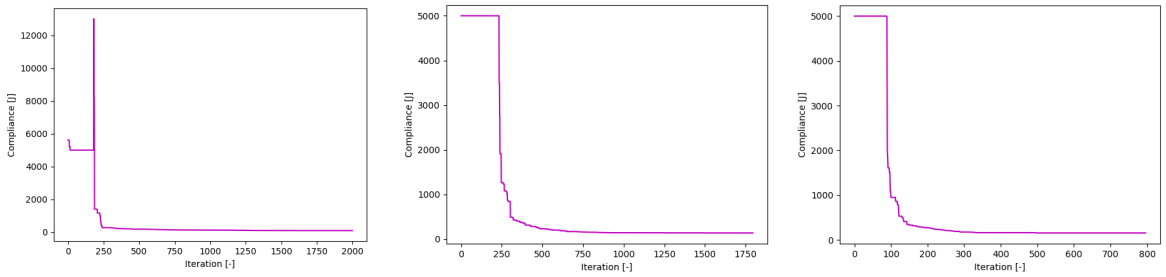
(a) *random* (b) *solid* (c) *triangle*

Figure 17: Convergence Trends for $f = 0.6$



(a) *random* (b) *solid* (c) *triangle*

Figure 18: Convergence Trends for $f = 0.5$



(a) *random* (b) *solid* (c) *triangle*

Figure 19: Convergence Trends for $f = 0.4$