Optimization of Genetic Algorithm Driven Web Design

Jordan Sinclair

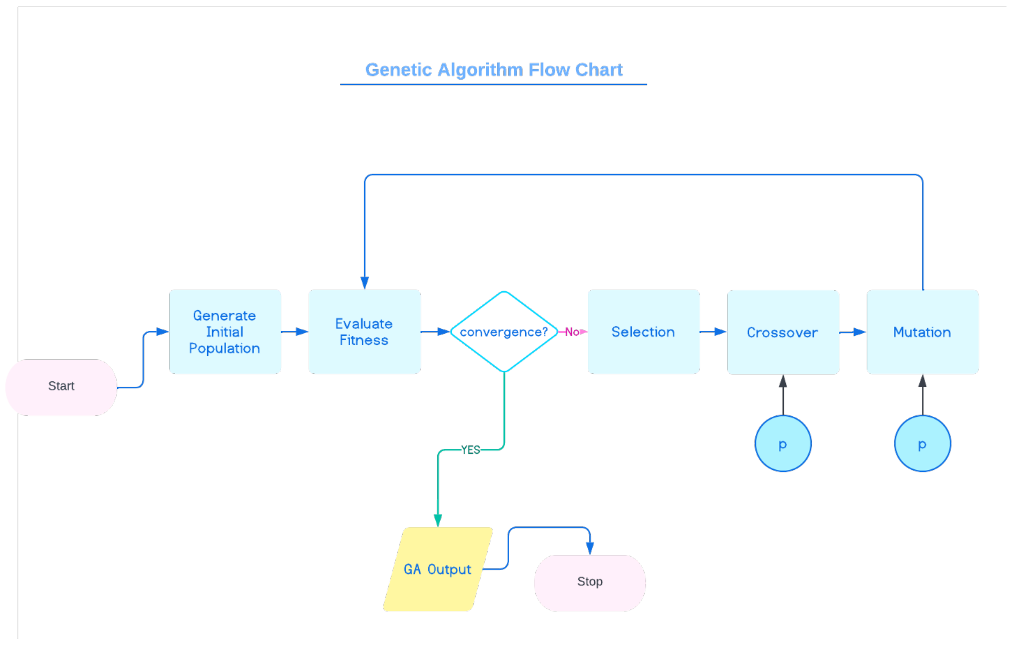
1 Introduction

Genetic algorithms offer a unique, biologically based, heuristic method for optimizing a large variety of problems. This method uses the ideas of genetics to perpetuate a "genome" through a series of evolutionary changes. In this paper, I describe a unique approach to the application of genetic algorithms (GAs) to web design via integer optimization. I hope to use integer programming techniques to constrain the genetic algorithm based on accessibility and layout considerations and ideally maximize the overall fitness of each population of websites. I plan to use the genetic algorithm as a heuristic input to the objective function of the optimization problem. The goal of this project is to determine if and to what extent genetic algorithms can be used for creative tasks such as web development and how, through solving the optimization problem, they might find unique and usable design patterns.

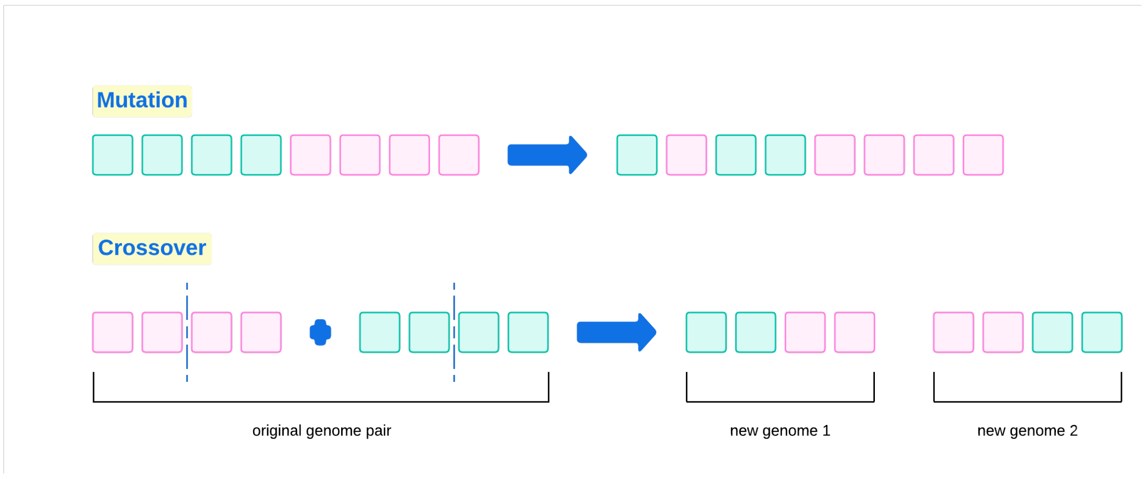
2 Background

Genetic Algorithms

Genetic algorithms are a type of search method consisting of a biologically based system in which a population of genomes is created and mutated over time. The "DNA" of the population is altered over time to produce new results. GAs implement randomized heuristics to allow for more unique and possibly more optimized genomes to occur. These heuristics typically take the form of mutation and crossover where a selected pair of the fittest genomes is used to create new genomes in the population. Thus, new genomes are created based on the fittest genomes of the previous population. The following image indicates the iterative process of a genetic algorithm.



Mutation changes one or more aspects of a given genome. So, for example, if the genome in question is a binary string, "110," the mutation process might flip one of the bits. We might then get "100." Crossover is a simple process of taking part of one genome and adding it to the corresponding part of another and vice versa. So, for example, we might have "111" and "010," leading to the crossover results of "011" and "110." The following image graphically depicts this process for reference.



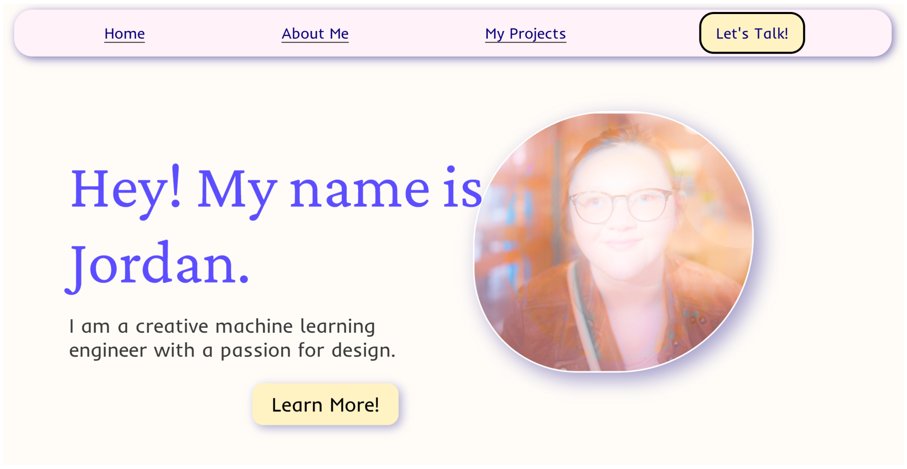
Integer Programming & Optimization

Integer programming is a subset of optimization that limits variables to integer or binary types while subjecting the function The idea is to minimize some type of cost function while meeting certain constraints. One problem that integer programming is commonly used for is something known as set packing. Set packing essentially ensures that all areas are covered by some set, I. For example, consider a cell service network system. We want to minimize the cost of building cell towers in the system but ensure that everyone has access to cell service.

Another typical example in this field is called a "knapsack" problem in which a hiker must fit four differently weighted objects in a backpack, but can only carry a certain amount. It is a question of cost versus utility in which we hope to maximize utility while meeting the cost constraints (which are weights in the hiker example). This is accomplished through a mathematical formulation of the problem by determining what form of constraints are necessary and how best to maximize the utility.

Web Design

Consider the image below that represents a fairly typical hero section layout; where a hero section is the attention grabbing part of a landing page. This is the section responsible for bringing business to companies, or, as is the case below, hiring managers to applicants.



Hey! My name is Jordan.

I am a creative machine learning engineer with a passion for design.

As you can see, the page is essentially broken up into columns with text on the left and an image on the right. Further, a traditional placement of the navigation bar is along the top of the page. Further, the text contrasts well with the background and is thus readable and accessible. There are specific requirements for text contrast that fall into a larger collection of accessibility requirements constituted by the Web Content

Accessibility Guidelines (WCAG). I will provide a more detailed mathematical discussion of this guideline when I consider the optimization constraints below, but it is essentially a ratio of the lightness of each color.

3 Literature Review

The design of unique websites through genetic algorithms is a fascinating topic and can be approached through several different lenses. Oliver, Monmarché, and Venturini suggest the use of a kind of interactive genetic algorithm in their article, "Interactive Design of Web Sites With a Genetic Algorithm" (). They generate a relatively small population and ask users to evaluate which websites they like the best. This feedback acts as an input to the fitness function so that the most liked (and most fit overall) websites can continue to create the next population through crossover and mutation. This allows the generated web designs to be more targeted towards the individual using the program.

In support of this, Oliver, Monmarché, and Venturini also broke up the web design into two facets: style optimization and layout optimization. Each component used its own genetic algorithm to generate new results so that the user could use both or only one depending on their needs. In terms of style optimization, a genetic code was created that included text colors and background colors for each object, font considerations such as font face, boldness, and size, and the text alignment. The authors incorporated luminance in their fitness rating of this genome. This allowed them to check the contrast between each elements text color and background color pairings.

For the layout optimization, a genome consisted of essentially an HTML table element. This allows different text and image objects to be placed in different rows and columns on the page. This allowed for constraints on the table such that the table was a certain size and had to be filled. Further constraints were also placed on the objects based on how many could be in a table cell and of what type. The authors found some interesting results through this method and determined that their users were benefiting from the interactive style of the fitness function. Oliver, Monmarché, and Venturini suggested the addition of constraints relating two or more objects to each other. They explained that this could indicate an issue with using both an interactive evaluation and an internal fitness evaluation.

Mensch suggested a similar interactive approach to genetic algorithm generated web designs in his article, "Optimizing Website Design Through the Application of an Interactive Genetic Algorithm" (), but included a kind of starting point for iterative process. He used a news article site as an example input to the genetic algorithm that consisted of 12 news articles with images and titles. The page also had a top navigation bar. He also used an HTML grid to indicate placement of objects in his site. Like Oliver, Monmarché, and Venturini, Mensch also considered both style and layout optimization, but combined them to a single genome rather than using two genetic algorithms.

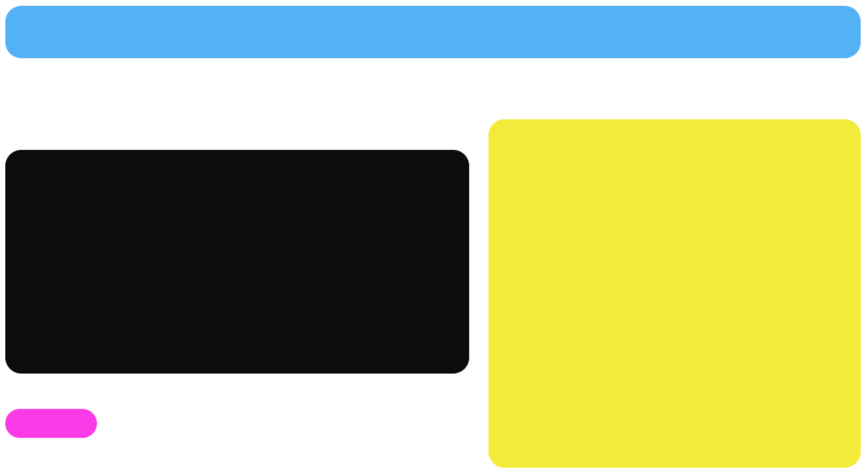
This genome consisted of options for the width of each object (how many columns of the table an object uses), and the background color, box shadow, font family, weight, size, and font color of both the navigation bar and the content. The content also include a style feature to determine if overlapping is allowed or not. Each component of the genome has only a specific set of values available. For example, the object width can only be 3, 4, or 5 columns in the table. This method seemed to generate favorable design patterns through user input, but the authors did suggest incorporating other metrics into the fitness function such as conversion rate or the time users spend on a page.

Günthermann and Kingston took a more mathematical approach to the evaluation of a web design in their article, "Designing a Website using a Genetic Algorithm" (). They considered elements such as symmetry, the Golden Ratio (a design rule relating to the size of objects on the page), and clarity. The authors also included color contrast in their fitness assessment, similarly to Oliver, Monmarché, and Venturini's approach to style optimization. Günthermann and Kingston used the relationship of objects one either side of the page to indicate symmetry. In their case, this included a side bar compared with images and text on the opposite side of the page. The weight was essentially the combined area of the page elements. Thus, they provided the following formula.

Additionally,

4 My Approach

I decided to use a combination of the approaches above in my genetic algorithm. I decided not to use a restricted initial layout in the design, but rather have specific components that will be randomly generated to any size anywhere on the page. I created image, button, text, and navbar classes in the hopes of finding an interesting hero section layout as described in the introduction. Each element can have a width and height up to the page size. For simplicity, I decided to create each component as a simple rectangle using pygame. The output will have an appearance similar to the image below. The color of the text will vary based on contrast with the white background.



Genome Creation

A genome is a single website that consists of the four elements discussed above (a button, navbar, some text, and an image). Each genome consists of a linked list of the page elements and their attributes.

Fitness Function

Using this as the basis for the genetic algorithm based optimization problem, we can start to form a fitness function. Consider the set of website components below which relates to each of the four rectangles shown above.

$$\textbf{Set } C = \{\textbf{navigation},\ \textbf{button},\ \textbf{image},\ \textbf{text}\}$$

Each component, C_i in the set corresponds to a set of attributes including the x, y position and the height and width of the rectangle.

$$\textbf{Set } A = \{\textbf{height},\ \textbf{width},\ \textbf{x},\ \textbf{y}\}$$

Thus, let the variable, $X_{i,j}$ represent the integer value of attribute j for component i. Now consider the constraints applicable to each component. First, consider the navbar component. It is desired that the x-coordinate of the navbar be towards the left side of the screen, which has a maximum width given by W. So, the first constraint should be:

$$X_{0,2} \leq 0.05W$$

I also wanted the y coordinate of the navigation to be close to the top of the page, so I added the following constraint.

$$X_{0,3} \leq 0.1H$$

Further, the width of the navbar should take up most of the page. Consider, then,

$$0.95W \leq X_{0,1} \leq W$$

Next, we want a reasonable maximum value for the height also proportionate to the page height, H. Thus, we add the constraint:

$$X_{0,0} \leq 0.1H$$

Finally, in order to maintain a horizontal rectangle, I added a constraint in which the width must be greater than the height of the navbar.

$$X_{0,1} > X_{0,0}$$

Next, consider the text component. There are three main constraints necessary for reasonable placement of the text. First, the x location should be towards the left side of the page. So,

$$X_{3,2} \leq 0.02W$$

Additionally, the width should be at least half the width of the page for readability. Thus, the folllowing constraint is added.

$$X_{3,1} \leq 0.5W$$

Finally, the height of the text should be at least about a third of the height of the page.

$$X_{3,0} \geq 0.3H$$

Now, consider the image component. The width of the image should be at least 40% of the size of the window, giving:

$$X_{2,1} \geq 0.4W$$

Further, the x coordinate must be within some range of the right side of the page. So,

$$X_{2,2} \geq 0.5W$$

Similarly, the height of the image should be at least a third of the height of the page.

$$X_{2,0} \geq 0.3H$$

Finally, there are two constraints for the button's width and height. First, I wanted the width to be between:

$$0.05W \leq X_{1,1} \leq 0.5W$$

Further, the height of the button should be less than 10% of the page height.

$$X_{1,0} \leq 0.1H$$

I also decided to include a few constraints that relate two or more components of each website together, respectively. The first concerns the overlapping of the image and text. Essentially, the difference between the x coordinates should be greater than 0 (no overlap). I think that it would be interesting to experiment with a Euclidean distance in the future to see if more accurate results could be achieved. Currently, I just added the following constraint.

$$X_{2,2} - X_{3,2} \geq 0$$

Next, I wanted the button to be under the text, so their y coordinates should be related by the following formula.

$$0.05W \leq X_{3,3} - X_{1,3} \leq 0.1W$$

Finally, I decided to implement the Golden Ratio formula between the image size and the text size. The Golden Ratio explains that the two elements should have a 1:1.68 ratio. So, consider the following constraint.

$$1.68X_{3,1}X_{3,0} - 0.02W \leq X_{2,1}X_{2,0} \leq 1.68X_{3,1}X_{3,0} + 0.02W$$

Therefore, the following integer optimization model can be developed using information from the genetic algorithm to calculate the fitness of each generation (which we essentially hope to maximize based on the constraints). Each constraint will either penalize or reward the objective function depending on if the constraint is met.

$$\max fitness$$

$$\text{s.t.: } X_{0,3} \leq 0.1H \tag{1}$$

$$X_{0,2} \leq 0.05W \tag{2}$$

$$0.95W \leq X_{0,1} \leq W \tag{3}$$

$$X_{0,0} \leq 0.1H \tag{4}$$

$$X_{0,1} \geq X_{0,0} \tag{5}$$

$$0.05W \leq X_{1,1} \leq 0.5W \tag{6}$$

$$0.01H \leq X_{1,0} \leq 0.1H \tag{7}$$

$$X_{2,1} \geq 0.4W \tag{8}$$

$$X_{2,2} \geq 0.5W \tag{9}$$

$$X_{2,0} \geq 0.3H \tag{10}$$

$$X_{3,2} \leq 0.02W \tag{11}$$

$$X_{3,1} \leq 0.5W \tag{12}$$

$$X_{3,0} \geq 0.3H \tag{13}$$

$$X_{2,2} - X_{3,2} \geq 0 \tag{14}$$

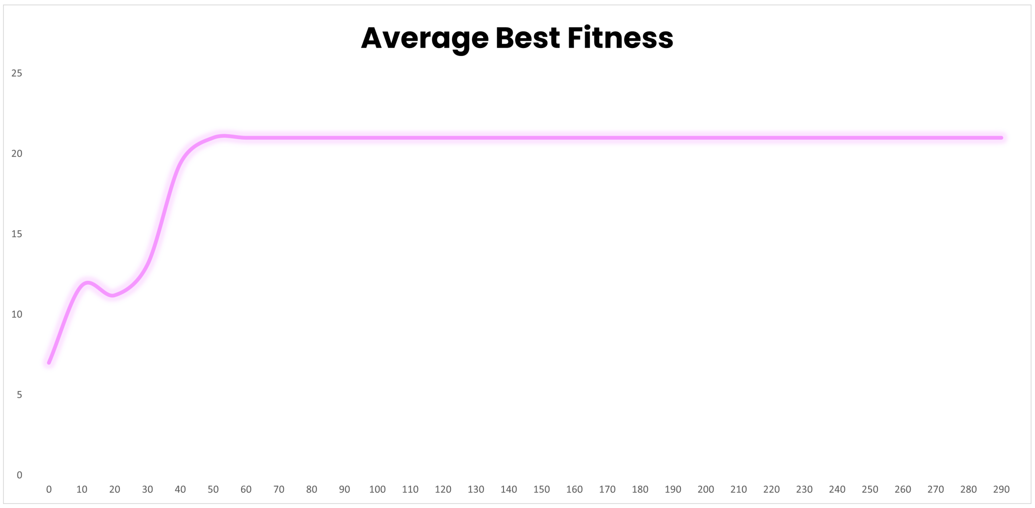
$$0.05W \leq X_{3,3} - X_{1,3} \leq 0.1W \tag{15}$$

$$1.68X_{3,1}X_{3,0} - 0.02W \leq X_{2,1}X_{2,0} \leq 1.68X_{3,1}X_{3,0} + 0.02W \tag{16}$$

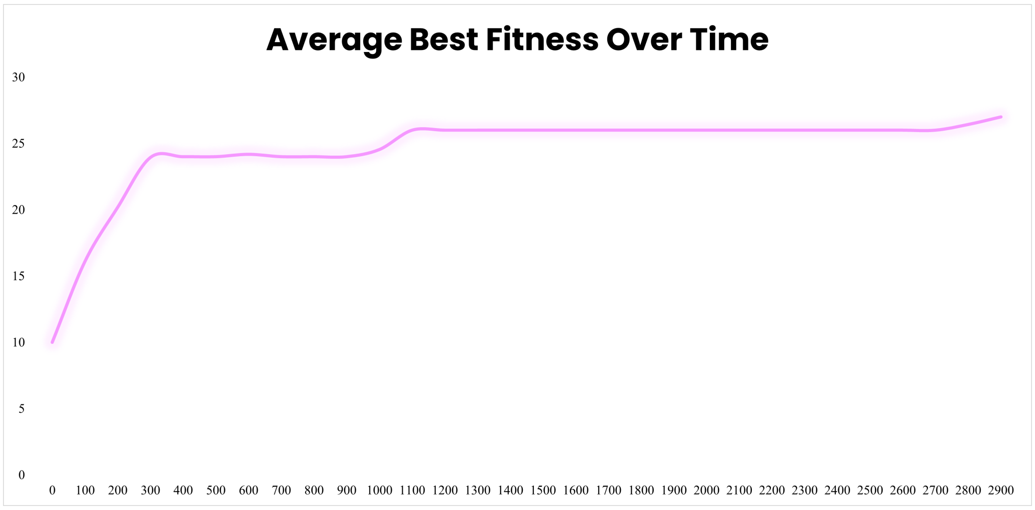
$$X_{i,j} \geq 0 \text{ integer } \forall i \in c, \forall j \in A \tag{17}$$

5 Experimentation

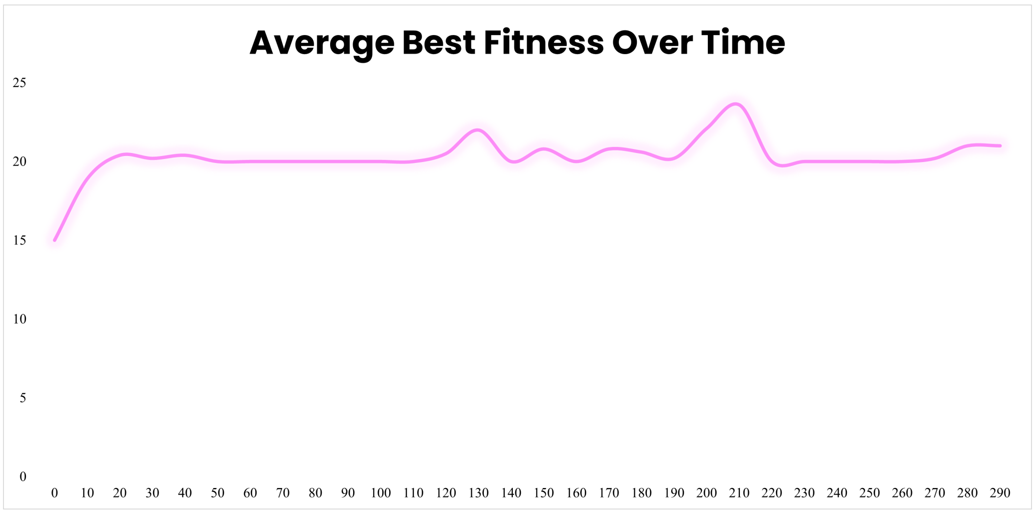
The following section contains various experiments with different population sizes, number of generations, and mutation and crossover rates to find an optimal arrangement of parameters. To start, consider an internet with 12 websites. With 300 generations, the following graph can be created depicting the average fitness over the generations. Each data point is the average of 10 best fitness values. The fitness seems to converge rather quickly. This could be good if the point of convergence is optimal.



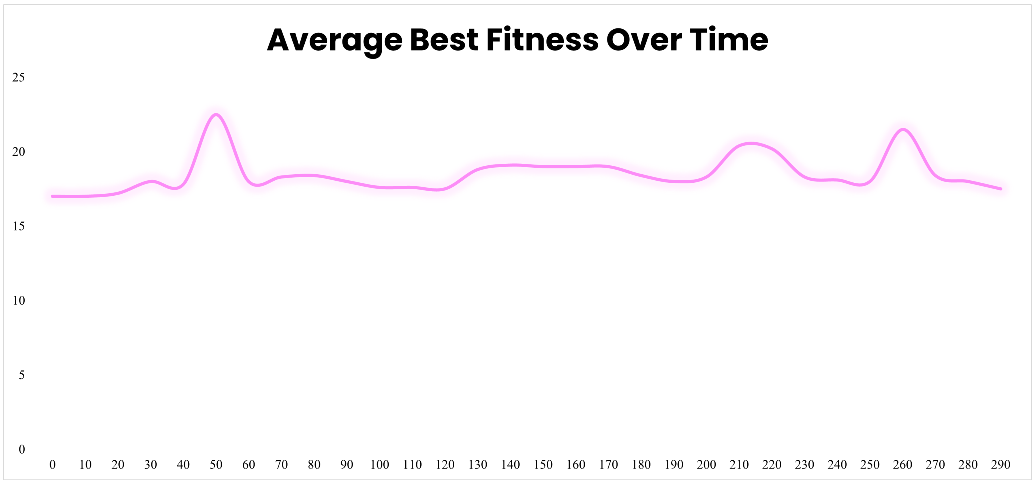
To see if this convergence was truly optimal, I next decided to try running the genetic algorithm with 3000 generations. The following graph shows the output of this trial in a similar manner as above. The best fitnesses are averaged every 100 generations in this case. The best fitness did reach a better optimum at about 27, compared with about 20 on the previous run.



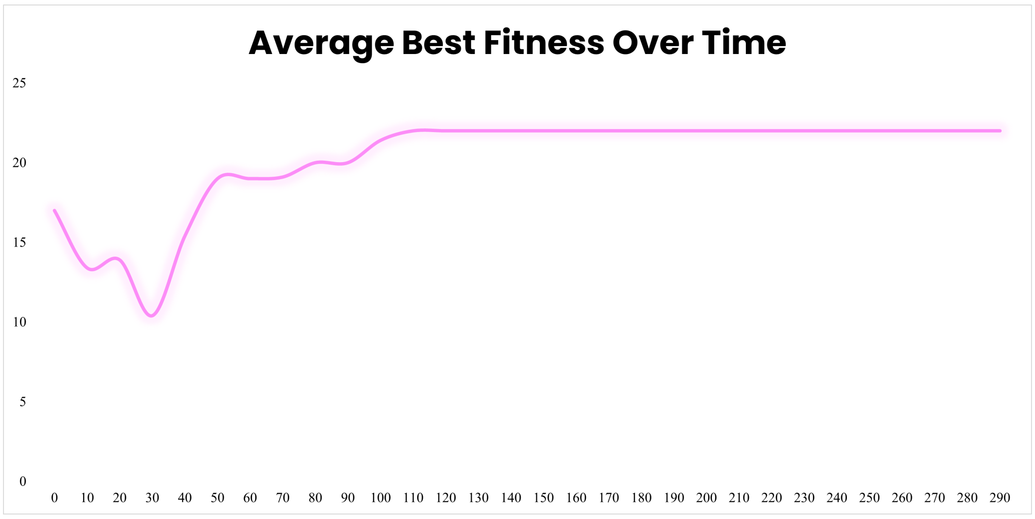
I also wanted to vary the size of the initial population to see if a higher fitness could be achieved. Setting the number of generations back to the original 300, I first tried a population size of 36 websites. The following image shows the outcome of the average best fitness. The results of this experiment were interesting. The fitness was more varied over the 300 generations, but did reach a higher value than the previous population size of 12. However, it didn't really seem to converge in the amount of time allotted.



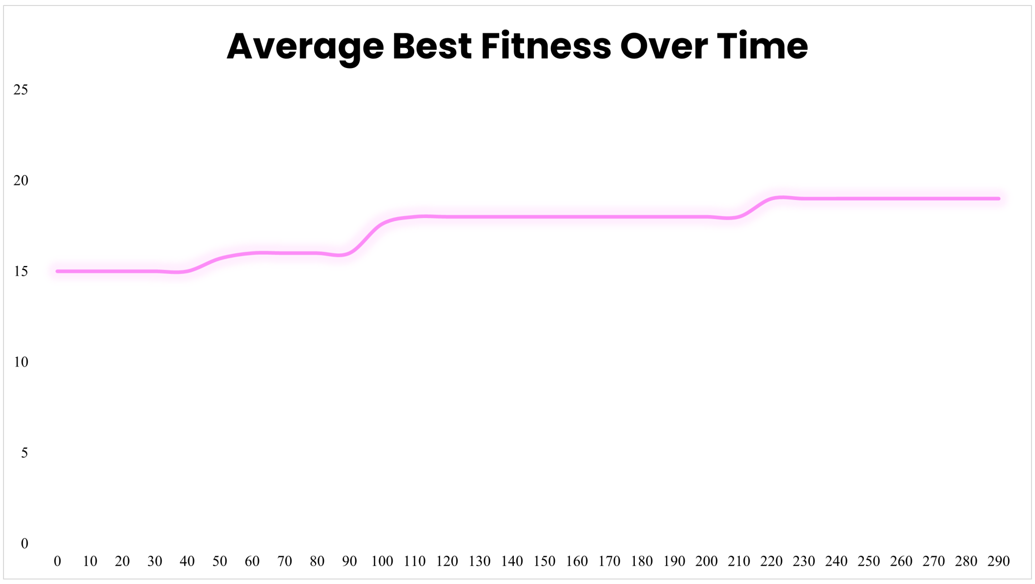
I decided to try a larger population size of 60 websites as well. This showed similar variance to the population of 36 websites, but the fitness scores were lower overall.



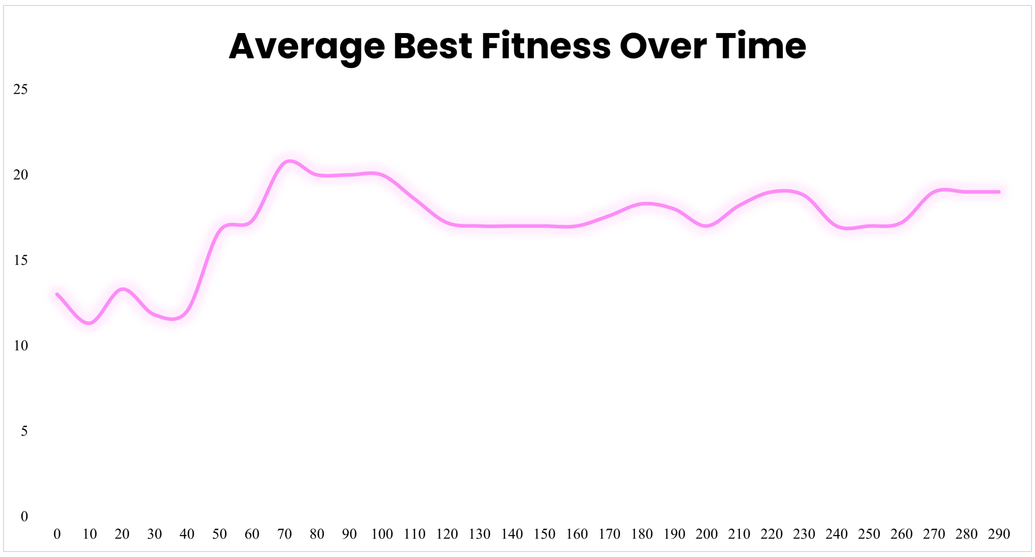
Setting the population size back to 12 websites, I changed the mutation rate to 0.1 and the crossover rate to 0.9. This did show convergence again at a slightly higher value than the original.



Next, I wanted to play with crossover and mutation a little more. I set the crossover value to 0.8 and the mutation to 0.01 to see what effect this might have on the fitness. As the following graph indicates, the average fitness increased in an almost stepwise manner, but remained lower than the original overall. This seems to suggest that the model was attempting to converge on suboptimal points at different stages in the process.

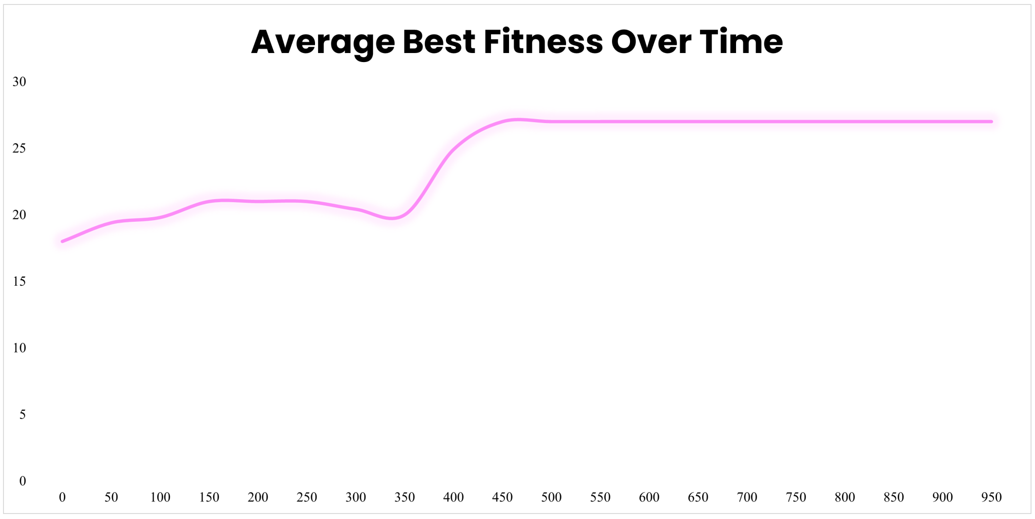


Returning to a more balanced ratio of crossover and mutation, I next decided to try a 75% to 25% mix of the two, respectively. This showed a lot more variance and didn't really seem to converge on any particular fitness value. Further, the average best fitnesses were significantly reduced as compared to the original or to the case with a 0.9 crossover rate and 0.1 mutation rate. Thus, it seemed like a mutation rate of 0.25 was too high given the scenario.



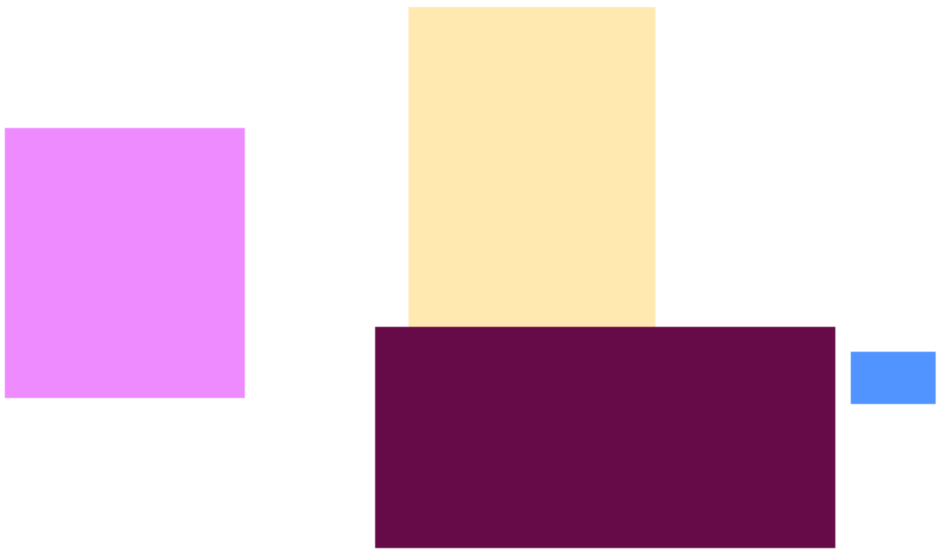
With the previous results in mind, I concluded that a crossover rate of 90% and a mutation rate of 10% seemed to be the best for this problem. It also seems like the number of generations should be around 1000. To test the population size further, I decided to use these parameters with a population size of 36 to see if any improvement could be seen. The following plot indicates the results of this. Of the results so far, it seems like this option performed the best. The highest average fitness found was 27, which the model converged on fairly quickly. It seems like this would be a reasonable balance between performance and complexity.

Finally, I also wanted to test the population size of 60 websites with the same parameters found above.

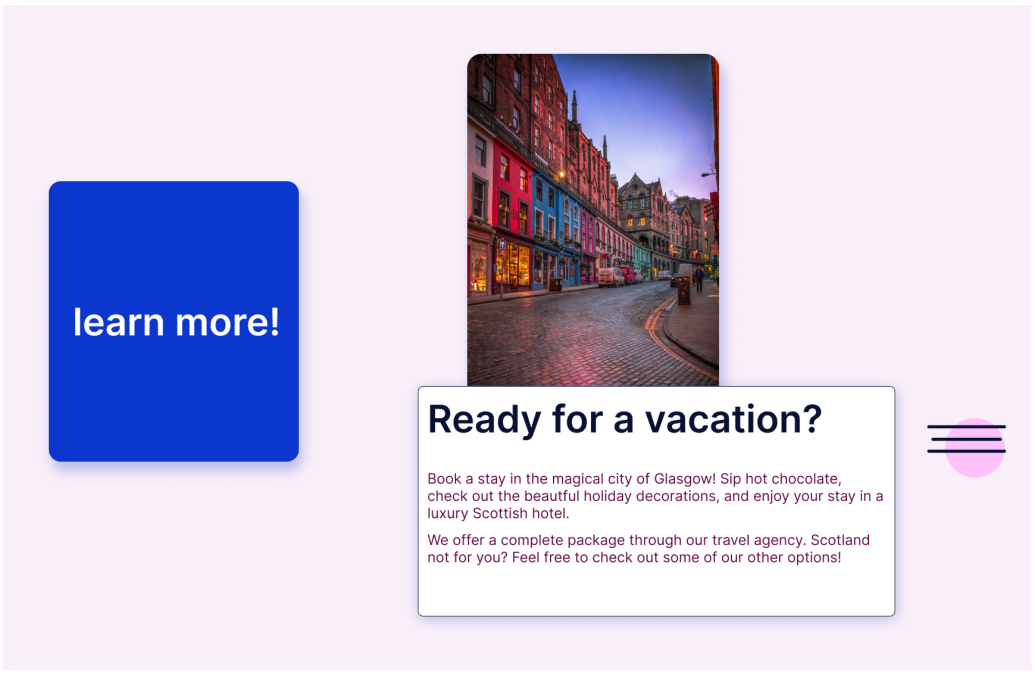


6 Results

The following image shows one of the outputs of running the genetic algorithm with a population size of 36 websites and a mutation rate of 0.1 and a crossover rate of 0.9. While this still isn't exactly what I was expecting in terms of the design, it does seem like something that could feasibly be used as a real website.



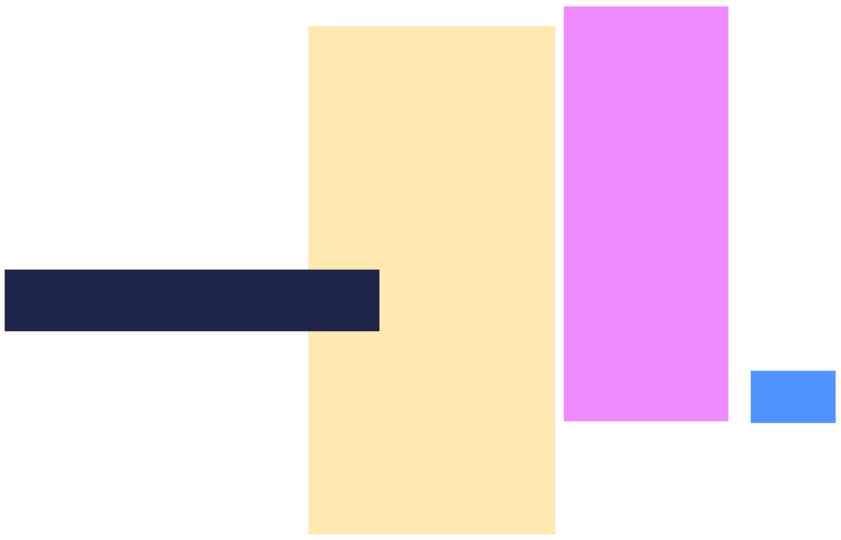
The pink rectangle is the button, the yellow rectangle represents an image, the blue rectangle is the navbar, and the maroon rectangle corresponds to text. The button is larger than usual, but the text and image are reasonable. They overlap in a unique way, which I think could lead to a more interesting layout. The navbar is too small to be a traditional desktop navigation, but would work as a hamburger menu that, when clicked, opens a full screen menu. I think that this could be used as a good starting point to build on. I created a quick representation of these rectangles as an actual web design using Figma (a design software). While I would still change the layout further, I think that this basic design shows promise. I used the same color generated for the text in the result below. I checked the contrast of the whole design using a plugin and found that all of the text was AAA rated (it contrasts very well). So, I think that the fitness of 25 was fairly indicative of the results. This image was found after 600 generations.



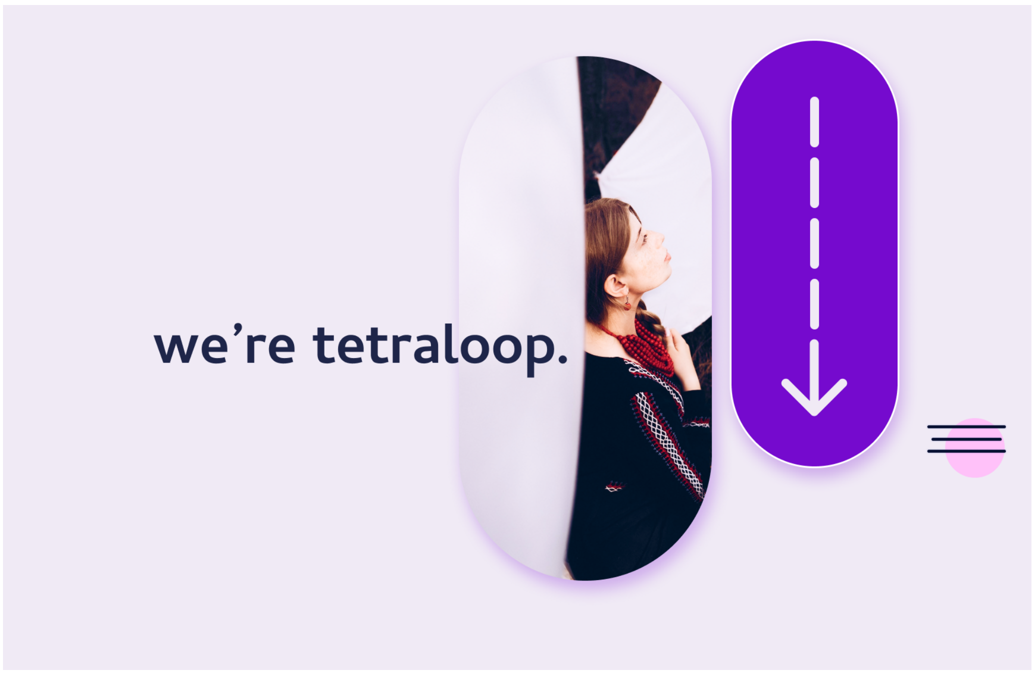
Book a stay in the magical city of Glasgow! Sip hot chocolate, check out the beautful holiday decorations, and enjoy your stay in a luxury Scottish hotel.

We offer a complete package through our travel agency. Scotland not for you? Feel free to check out some of our other options!

At the 800th generation of this run, the genetic algorithm created the following design. Similar to the one above, this seems like it could feasibly be interpreted as a usable website. There are similarities with the first example, suggesting the presence of mutation. The image and navbar seem to be in the same location, but the text and button mutated. The previous design did receive a slightly higher fitness value at 25, whereas this design was given a fitness of 24. However, I personally would prefer the latter design. This suggests the importance of potentially incorporating an interactive evaluation as many of the authors in the literature review suggested.



Like the previous version, I also created a more flushed out web design in Figma based on the generated layout. This is shown below. I think it turned out well, but there are definitely some interesting design decisions. Particularly, the button is very long and skinny. I wasn't sure how to interpret it at first. I thought about using vertical text, but it took too much attention away from the header and was difficult to read. I decided to use an arrow indicating the user to continue scrolling.



7 Concluding Thoughts & Future Work

The aim of this project was to determine the feasibility of using integer programming and genetic algorithms to create artistic and unique web designs.