Investigating the effectiveness of Genetic Algorithms in character strength-based grouping of people

Jiang Zhiheng, Alistair Cheong Liang Chuen, Low Wei Sheng, Wu Jiayang

Abstract Mounting empirical evidence indicates that well-planned teams are immensely productive. However, it is often difficult for humans to group themselves based on their character strengths, as such a grouping process is subject to biases such as personal preference and closeness between individuals. Hence, this study proposes an objective method to form optimal character strength-based groups using a genetic algorithm. The top 5 VIA character strengths of 2 classes of 25 students were collected and represented in a 6-dimensional vector form corresponding to the 6 broad categories of VIA character strengths. Subsequently, pairwise cosine similarities were calculated for each student in each class, and this data was fed through 3 separate fitness functions corresponding to 3 different definitions of what counted as an "optimal" grouping. A genetic algorithm was then utilized to generate numerous possible configurations of groupings, and the fittest configurations were selected for. Our genetic algorithm successfully grouped the students based on our 3 fitness function criteria, and yielded results that were significantly different from random groups. A poll conducted amongst 35 of the 50 students also showed that groups are the most effective when people with similar character strengths are together.

1. Introduction

Mounting empirical evidence indicates that well-planned teams are immensely productive. However, it is often difficult for humans to group themselves based on their character strengths, as such a grouping process is subject to biases such as personal preference and closeness between individuals. Hence, there is a need for an algorithm that can objectively determine the best character-based groupings, given a fixed number of people and their character strengths.

Genetic algorithms are often used in multicriteria problems where the phenotypes that should be optimized to obtain the optimal solution are either unknown or are too complex to be properly quantified for optimization. As the problem of grouping people in an optimal manner satisfies the above criteria, we believe that a genetic algorithm is the most suitable algorithm for optimizing this problem.

Hence, we propose a genetic algorithm that can optimally group people based on their top 5 character strengths. However, the definition of "optimal" is up for debate. An "optimal" group could be one whereby the members have similar traits, different traits, or traits that are neither too similar nor too different. Thus, we derived 3 variations of the genetic algorithm, with each one of them addressing a different "optimal" way of grouping people together.

2. Literature Review

Yang, Qinghong & Chen, Long. (2013) have derived a learning group algorithm with learning interest, learning capability, learning style, learning activity, sex, and age as the factors of personalities. However, the algorithm for this research paper is used to form learning groups, and not just groups in general. Although the experimental results verify the algorithm, there is no justification for the rationale behind their choice of optimization algorithm. **(not sure coz i didnt read the paper)**.

3. Hypotheses

In this study, it was hypothesized that:

- People prefer being in a group with others that possess personality traits similar to them; and
- A genetic algorithm can be used to determine an optimal configuration of groupings, given the character traits of people.

4. Assumptions

In this study, it was assumed that:

- A group is considered optimal according to 3 definitions:
    - Members with similar traits are grouped together;
    - Members with dissimilar traits are grouped together;
    - The group consists of a balanced mix of people with similar and dissimilar traits;
- No more optimal way of grouping exists as a middle ground; and that
- The compatibility of members in a group depends solely on their personalities.

5. Methodology

5.1. The 6D character strengths vector

The VIA Character Strengths Test profiles an individual using 24 character strengths from 6 broad categories[1]:

1. Wisdom
2. Courage
3. Humanity
4. Justice
5. Temperance
6. Transcendence

Hence, a 6D vector v was used to represent each student's character profile.

$$v = [S_1 \ S_2 \ S_3 \ S_4 \ S_5 \ S_6]$$

For each axis, the value S_i $(1 \le i \le 6)$ represents the student's score for the i^{th} character strength category. S_i is calculated by counting the number of character strengths from the i^{th} category that were present in the student's top 5 strengths. Thereafter, this count was divided by total number of character strengths in the i^{th} category. This produces a score for S_i, where $0 \le S_i \le 1$.

This was done to normalize the scores, because some categories of character strengths had more strengths than others. In doing so, each category of character strengths was treated equally for an unbiased mathematical representation of the student's character profile.

5.2. Cosine Similarity

A cosine similarity function was used to gauge the similarity between 2 students' 6D vectors. Cosine similarity measures the cosine of the angle between 2 vectors projected in a multi-dimensional space[2]. The smaller this angle, the greater the similarity between the 2 vectors.

The equation for cosine similarity is as per Fig. 5.2.1:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Figure 5.2.1: Full equation for cosine similarity. *$A \cdot B$ is the dot product of vectors A and B. $\|A\|$ and $\|B\|$ are the magnitudes of vectors A and B respectively. More specifically, the double pipe notation refers to the Euclidean norm, which is an extension of Pythagoras' Theorem applied to any n-dimensional vector. A_i and B_i are the components of vectors A and B respectively.*

The resulting similarity ranges from −1 (meaning the 2 vectors are exact opposites of each other), to 1 (meaning the 2 vectors are identical). A similarity of 0 indicates orthogonality or decorrelation, while values in between represent intermediate similarity or dissimilarity[3].

However, as mentioned in Section 5.1, $0 \le S_i \le 1$. Therefore, in this study, there were no negative cosine similarity values between students.

For each of the 2 classes involved in this study, the cosine similarity between every pair of students was computed and stored in an array to be used in the genetic algorithm. The similarities can then be visualized in a heatmap as per Fig. 5.2.2 and Fig 5.2.3.

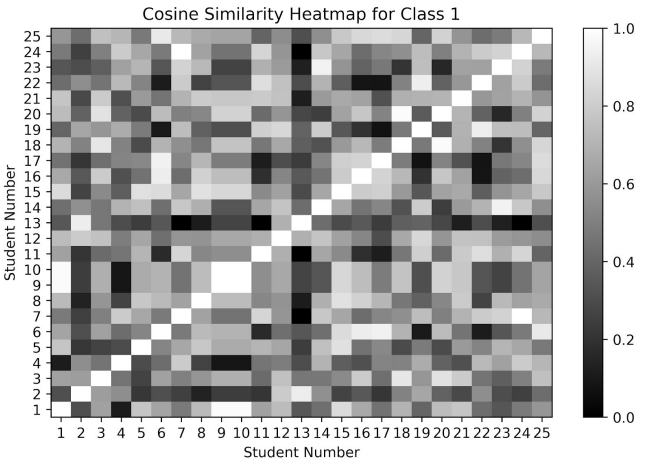

Figure 5.2.2: Cosine similarity heatmap for Class 1.

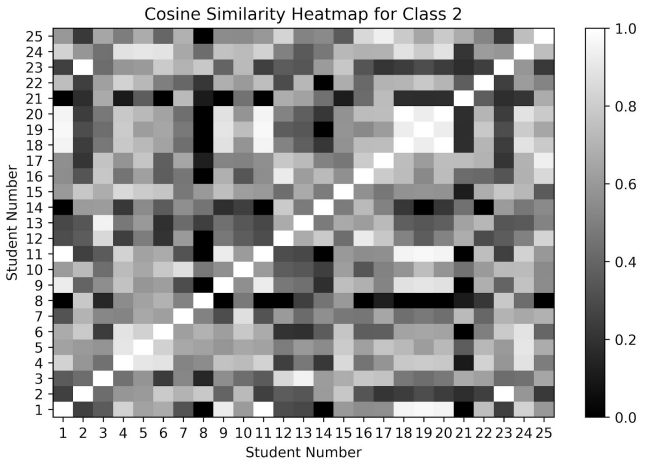

Figure 5.2.2: Cosine similarity heatmap for Class 2.

5.3. Fitness Functions

5.3.1. Definition of a Fitness Function

As this study aims to find a better way to group students, it is imperative that we define a metric to describe the desirability of a particular group.

Consider 5 groups consisting of 5 students made out from a class of 25 students. In this study, this is defined as a configuration. The fitness function then aims to measure how optimal this configuration is.

5.3.2 The 3 Fitness Functions

As what counts as an "optimal" way of grouping people is subjective as discussed in section 4, we derived 3 fitness functions to account for the intuitive optimal ways. The best fitness function will be found in section 5.5.

The first fitness function, f_1, assumes that the ideal group is one where the members are as different as possible. This fitness function therefore aims to form a configuration where the sum of the cosine similarity values of all possible pairs within the configuration is minimized.

The second fitness function, f_2, assumes that the ideal group is one where the members are as similar as possible. This fitness function thus aims to maximize the cosine similarity of all pairs in a configuration.

The last fitness function, f_3, assumes that the ideal group contains a balanced mix of people who are both similar to and different from one another. This fitness function hence aims to form a configuration such that the absolute sum of the cosine similarity value of all possible pairs within the configuration is minimised.

The functions used to compute our fitness function are as follows:

$\varepsilon_i =$ 1

5

$$\epsilon_i = \frac{1}{5}\sum S_{\mathrm{u,\ v}} \text{ for } i \in [1,\ 5]$$

$$\omega = \frac{1}{5}\sum \epsilon_j \text{ for } j \in [1,\ 5]$$

.

where $S_{u,v}$ refers to the cosine similarity between the uth and vth person in the group, ϵ_i refers to the average of cosine similarities of every pair in the ith group, and w refers to the average of the average of cosine similarities of all groups.

The three corresponding fitness functions are as follow:

$$f_1(\omega) = \omega$$

$$f_2(\omega) = -\omega$$

$$f_3(\omega) = |\omega|$$

5.4. Genetic Algorithm

5.4.1. Definition

A genetic algorithm is a search and optimisation algorithm inspired by Darwin's Theory of Natural Evolution[4]. Similar to how Darwin's Theory of Natural Evolution follows the survival of the fittest, this algorithm aims to optimise the fitness of a certain fitness function to optimise the solution to a problem.

5.4.2. Initial Population

An initial population of 50 configurations is generated at random. A population refers to the set of configurations at a particular stage. We experimented with different values of population sizes, and the population size of 50 was selected as conventional genetic algorithms often rely on a population size of around 100. As the number of people in each class was 26, we determined that the best number of people in a configuration would be 25, as we can form groups with the same number of people (i.e. 5 groups of 5). As our fitness function does not account for a varying number of people in groups in a configuration, we ensure reliability of our algorithm by eliminating this factor via making it a constant. However, due to our small sample size of 25 people, a larger population is not necessary. It is critical to emphasise that initially, all configurations and groups are determined randomly before the genetic algorithm is performed. This is also referred to as the first generation (refer to section 5.4.4).

5.4.3. Fitness Function

As mentioned in section 5.3, a fitness function measures how optimal a configuration is. Hence, the algorithm would aim to maximise this function to achieve an optimal configuration.

5.4.4. Generations

A generation refers to the stage at which the algorithm is at. For each generation, we perform 3 steps to generate the next generation: Selection, Crossover and Mutation, which will be elaborated in the next few sections. At the start and end of each generation, the population size (the number of configurations in the generation) remained the same, which, in this case, was 50. In this study, the algorithm progressed through 5000 generations. The average fitness of the configurations in each generation was expected to plateau at some maximum value after a certain number of generations. After this occurs, it would be determined that the optimal configuration of groupings had been achieved.

5.4.5. Selection Operation

At the beginning of each generation, all 50 configurations were sorted by their fitness values from the previous generation. The top 10% (5 configurations) of the previous generation was automatically advanced to the current generation, as they were determined as the fittest of the generation. Similarly, the bottom 10% was determined as the least fit in the previous generation, and were automatically eliminated, as they would be unlikely to be able to improve current configurations in the population.

5.4.6. Crossover Operation

Based on the top 50% of the population of the previous generation, two configurations were picked at random. The top 50% was chosen because the configurations possessed attributes which were above average for the population; hence these attributes, or 'genes', should be propagated to future generations. The 'genes', or characteristics, of these generations would be which group each person belongs to. Let the two configurations chosen be Configurations A and B. Let the set of people in these two configurations be S.

For each person in Set S, suppose person P is in Group X in Configuration A, and Group Y in configuration B. Note that these two groups can be the same. After the crossover, person P would be classified randomly into either Group X or Group Y. If the person cannot be classified under one of the two groups, then the person is automatically classified into the other group (for example, when one of the groups already has 5 members). This repeats until every person is classified into one of the two groups, which they were initially from. If the person cannot be classified into either group, the crossover operation is restarted.

The fitness of the new configuration would then be compared with the parent configurations. If the fitness of the new configuration is lower than that of the parent configurations, the crossover operation is restarted to obtain another configuration. If no possible better configuration is determined after multiple attempted crossovers, the better parent configuration would be selected to proceed to the next generation, as it would be considered already optimized.

In order to ensure that the order of the groups do not affect the results, it was randomly shuffled and the crossover step was repeated multiple times. Out of all the successful crossovers, the configuration

with the maximum fitness value based on the fitness function would be selected to proceed on to the next generation.

The crossover operation was performed to generate 90% of the remaining population, as the first 10% had already been determined by the selection operation described in section 5.4.5. In other words, the crossover operation was repeated 45 times for every generation.

5.4.7. Mutation Operation

At the end of each crossover operation, in order to ensure genetic diversity in our search space, a mutation operation was performed to vary the people in the different groups, to complement the randomly generated population. The mutation operation is performed on each configuration in the population. This is conducted by swapping the positions of two people in the configuration who are randomly chosen, which were from different groups. If the resulting configuration after the mutation yielded a higher fitness value, then the mutated configuration would be selected instead, replacing the initial configuration.

5.5. Questionnaire

A questionnaire was sent out to two classes, after groupings were for each class were generated (each class of 25 students). This allowed the identification of the preferred fitness function, and also the confirmation of the quality of the groups generated. This was accomplished by asking respondents to rank a few groupings, in terms of how well they think they can work with the group. In each question, one group was generated by each of the fitness functions. A fourth, randomly generated group was added as well.

The best fitness function was determined by finding out which one was consistently chosen as the best group. The quality of the groups generated was confirmed using the randomly generated grouping. If the score of the best fitness function is above that of the random grouping, it would show that the function was not chosen by chance, and the algorithm was the reason behind its success.

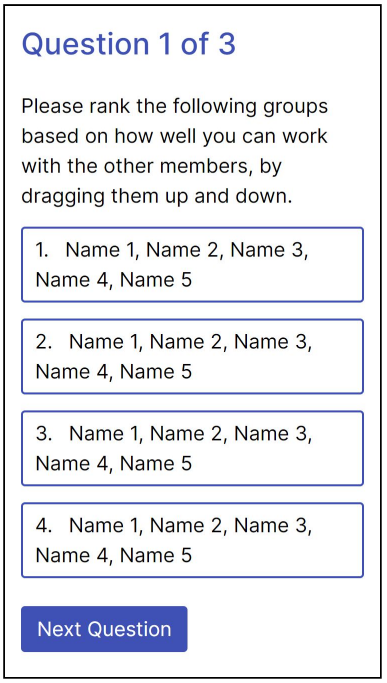

Figure 5.5.1. *Sample screenshot of the form sent out to respondents.*

This questionnaire was presented in a manner, such that respondents were unaware of the fitness function used to generate each grouping, to minimize bias. Each person submitted a response consisting of three rankings.

The score for each fitness function was determined by counting the number of rankings that chose that fitness function as the top choice. The lower rankings were not considered.

6. Results & Discussion

To aid data visualization, the fitness functions f_i $(1 \leq i \leq 3)$ will hereon be referred to using the genetic algorithm modes “DIFFERENT”, “SIMILAR” and “NEUTRAL” respectively.

6.1. Fitness vs. Generation

The average and best fitness of each generation was tracked for a total of 5000 generations. Min-max normalization was then applied on the fitness values to compress them to the range of 0 to 1 for easy viewing as per the following formula:

$$f_{normalized} = \frac{f_{original} - f_{min}}{f_{max} - f_{min}}$$

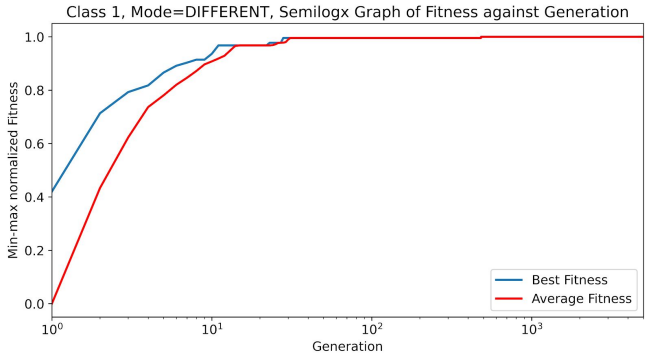

Figure 6.1.1. Semilogx Graph of Fitness against Generation for Class 1 when students of different character traits were grouped together.

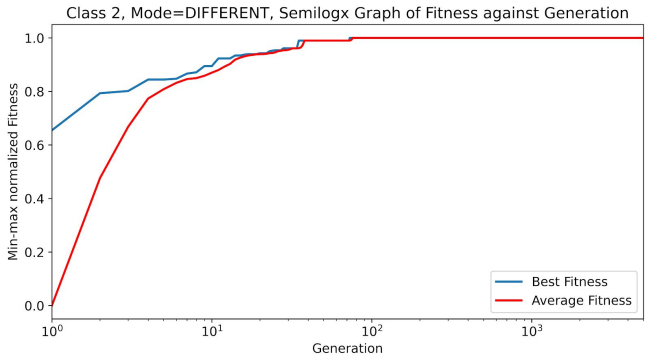

Figure 6.1.2. Semilogx Graph of Fitness against Generation for Class 2 when students of different character traits were grouped together.

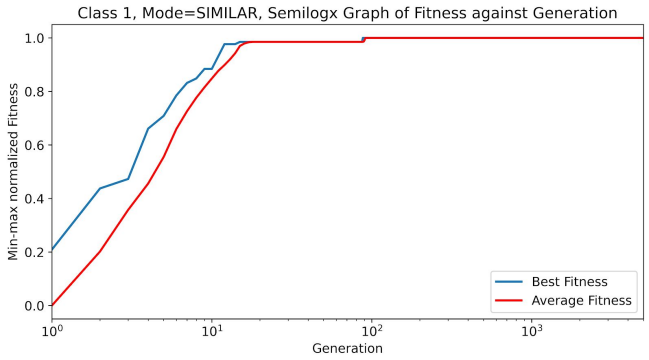

Figure 6.1.3. Semilogx Graph of Fitness against Generation for Class 1 when students of similar character traits were grouped together.

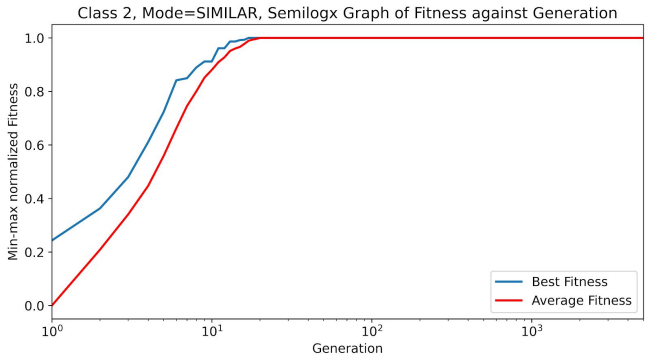

Figure 6.1.4. Semilogx Graph of Fitness against Generation for Class 2 when students of similar character traits were grouped together.

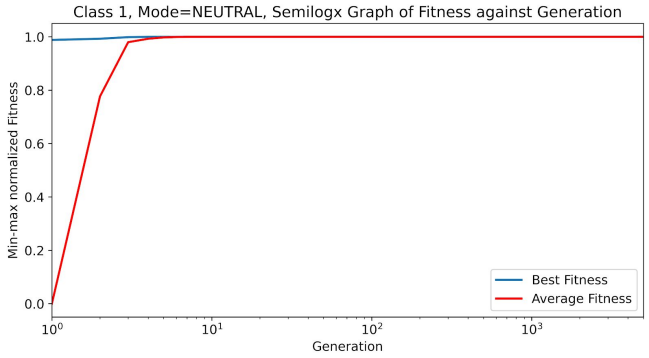

Figure 6.1.5. Semilogx Graph of Fitness against Generation for Class 1 when a mix of students with similar and different character traits were grouped together.

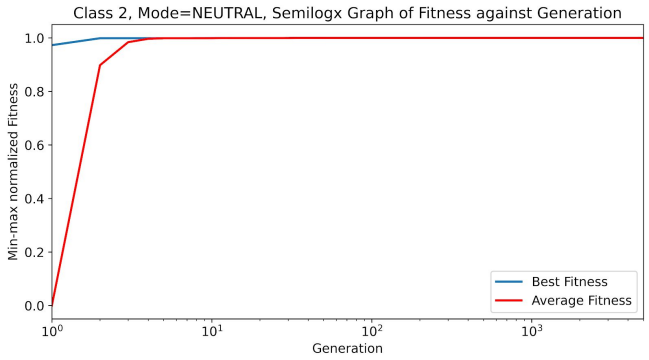

Figure 6.1.6. Semilogx Graph of Fitness against Generation for Class 2 when a mix of students with similar and different character traits were grouped together.

As seen from Figures 6.1.1-6, for each mode, the genetic algorithm was successful in optimizing their fitness values such that they approached 1. By looking at the horizontal scale, we can conclude that a near-optimal grouping is achieved by the genetic algorithm in around 100 generations or so.

Interestingly, the genetic algorithm was able to achieve extraordinarily high fitness for mode=NEUTRAL even in the first 10 generations. This could have happened as the initial random configurations generated by the genetic algorithm could have already possessed a decent mix of students with both similar and different traits. As a result, the fitness was high from the start.

6.2. 3D grouping visualization using PCA

PCA (Principal Component Analysis) is a technique to reduce the dimensionality of an n-dimensional dataset. It represents the original set of n variables using n *principal components*. Each principal component is a mixture of some of the original variables, and every principal component is decorrelated to each other. Majority of the

information in the original dataset is represented in the first principal component, with each subsequent principal component containing progressively lesser portions of such information. Hence, to reduce the dataset's dimensions from n to k, only the first k principal components are selected and plotted.

Such a method was required to visualize the groupings of students, as the 6D character strength vector could not be plotted on a graph. As such, PCA reduction was employed to reduce the 6D character strength vector to a 3D space. A 3D space was chosen over a 2D space as it had higher information retention, i.e. more information from the original 6D vectors was preserved when the data was reduced to a 3D space. More details on how PCA reduction works in our case can be found in Section 9.2.2.

In the following 3D plots, *information retention* refers to the amount of information that PCA reduction managed to preserve while performing dimensionality reduction. *Inertia* is a measure of the size of the spread of the students' data points, with a higher inertia meaning a higher spread. Further details on inertia will be expounded in Section 6.3.

PC1, PC2 and PC3 refer to principal components 1, 2 and 3 respectively. The various data points of the students were demarcated with their respective student numbers ranging from 1 to 25. To facilitate easy viewing of groups as 3D structures, the 3D convex hulls formed by the various groups' data points were shaded along with their centroids marked out as yellow circles. The 3D convex hull's size is an indication of the character trait spread.

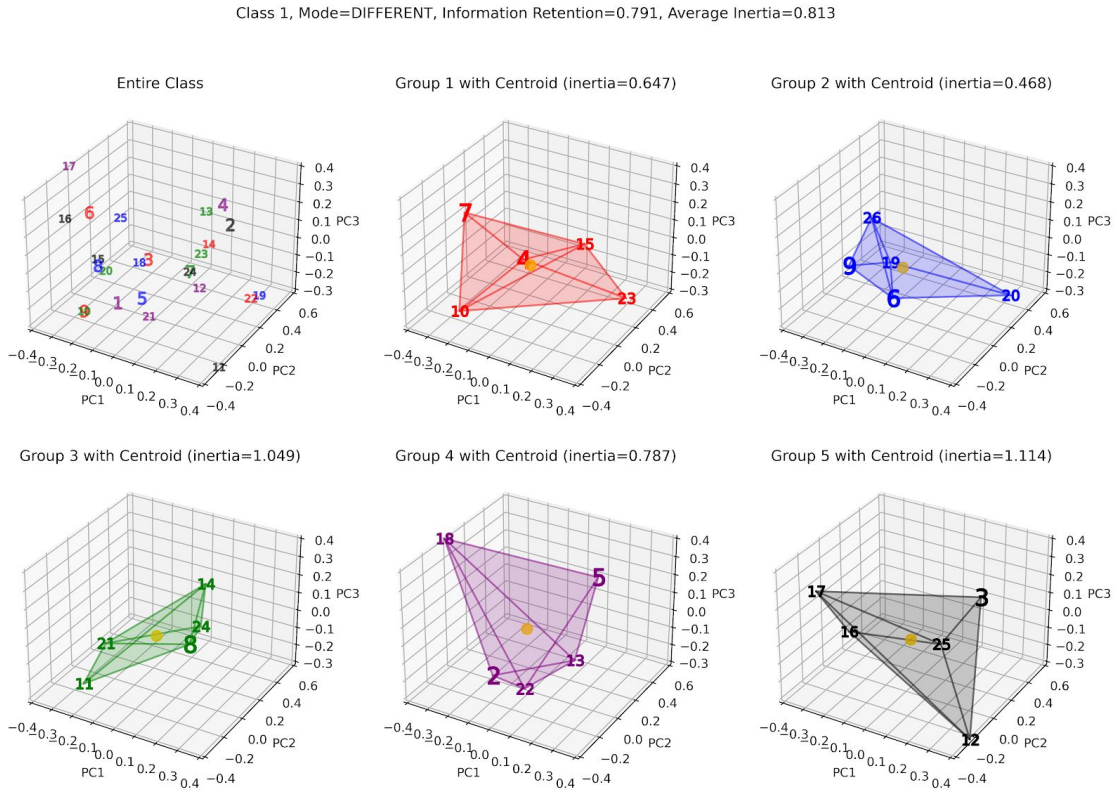

Figure 6.2.1. PCA reduced 3D visualizations of the entirety of Class 1, along with Groups 1-5 when students of different character traits were grouped together.

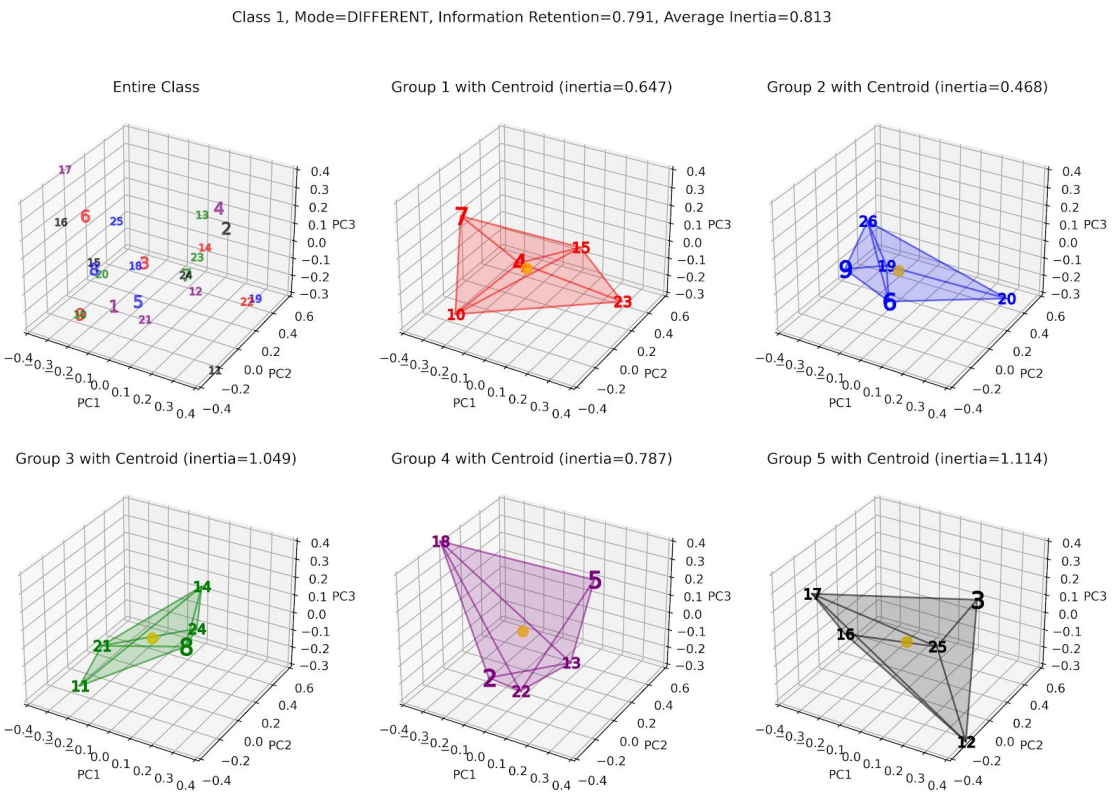

Figure 6.2.2. PCA reduced 3D visualizations of the entirety of Class 2, along with Groups 1-5 when students of different character traits were grouped together.

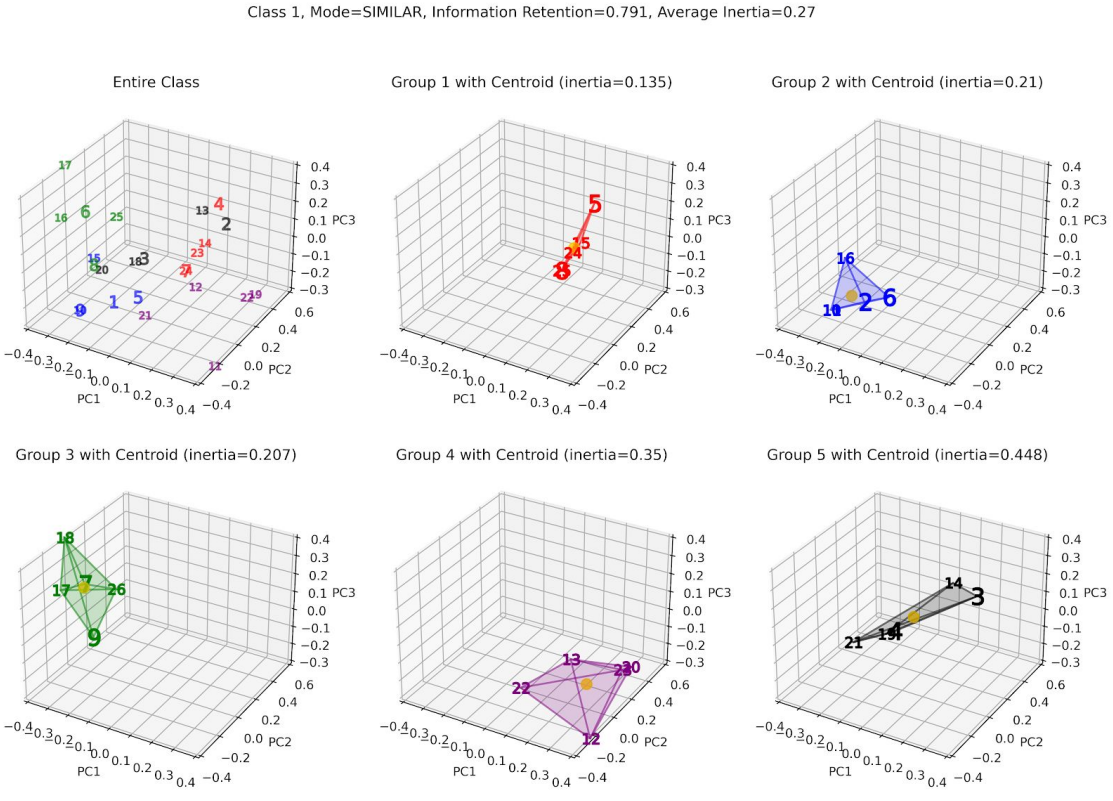

Figure 6.2.3. PCA reduced 3D visualizations of the entirety of Class 1, along with Groups 1-5 when students of similar character traits were grouped together.

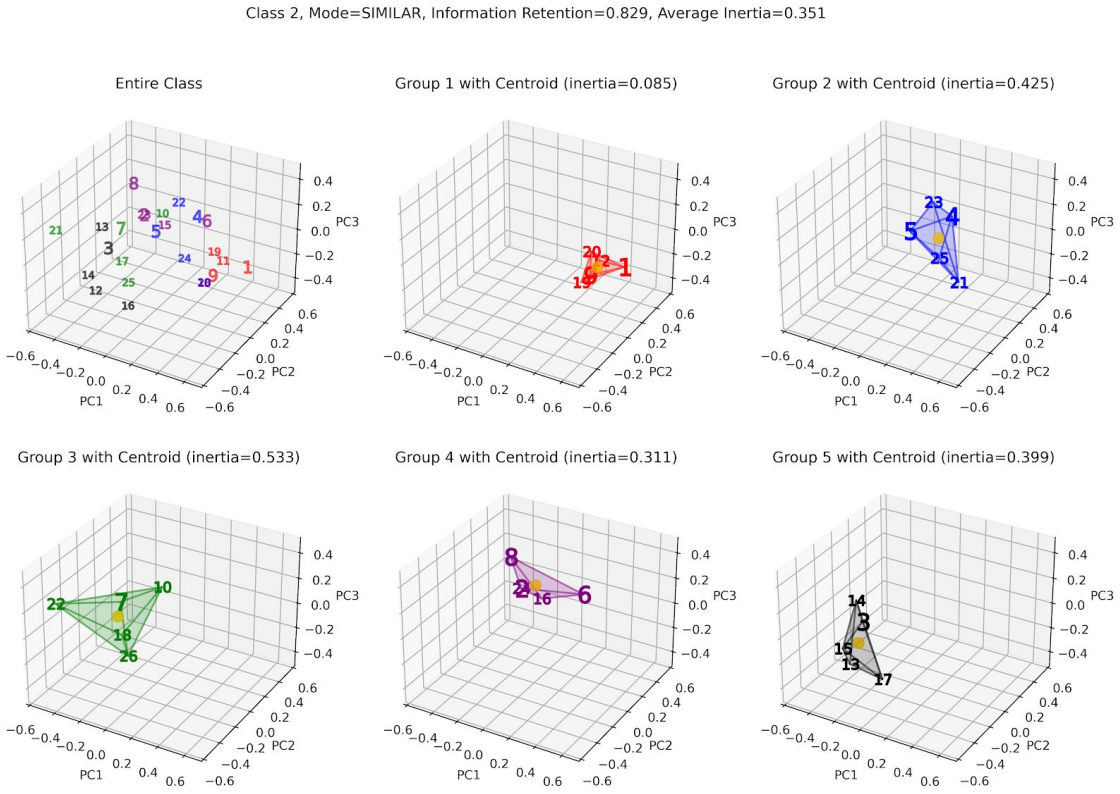

Figure 6.2.4. PCA reduced 3D visualizations of the entirety of Class 2, along with Groups 1-5 when students of similar character traits were grouped together.

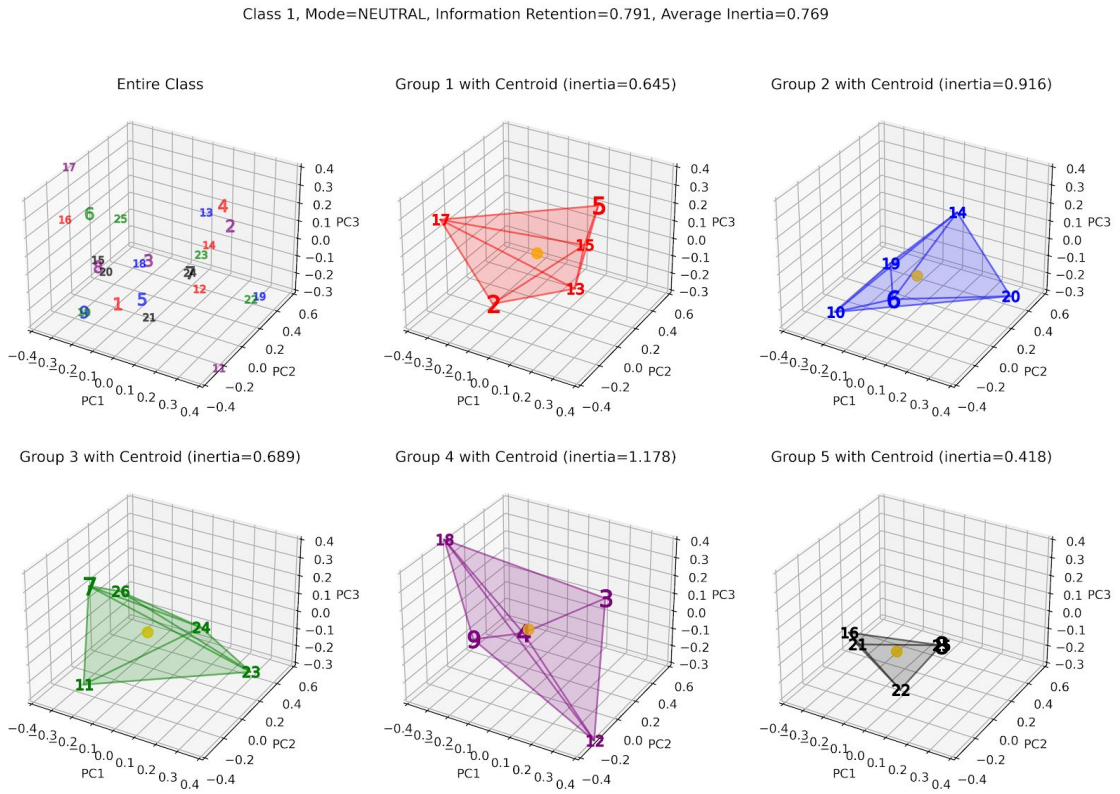

Figure 6.2.5. PCA reduced 3D visualizations of the entirety of Class 1, along with Groups 1-5 when a mix students with both similar and different character traits were grouped together.

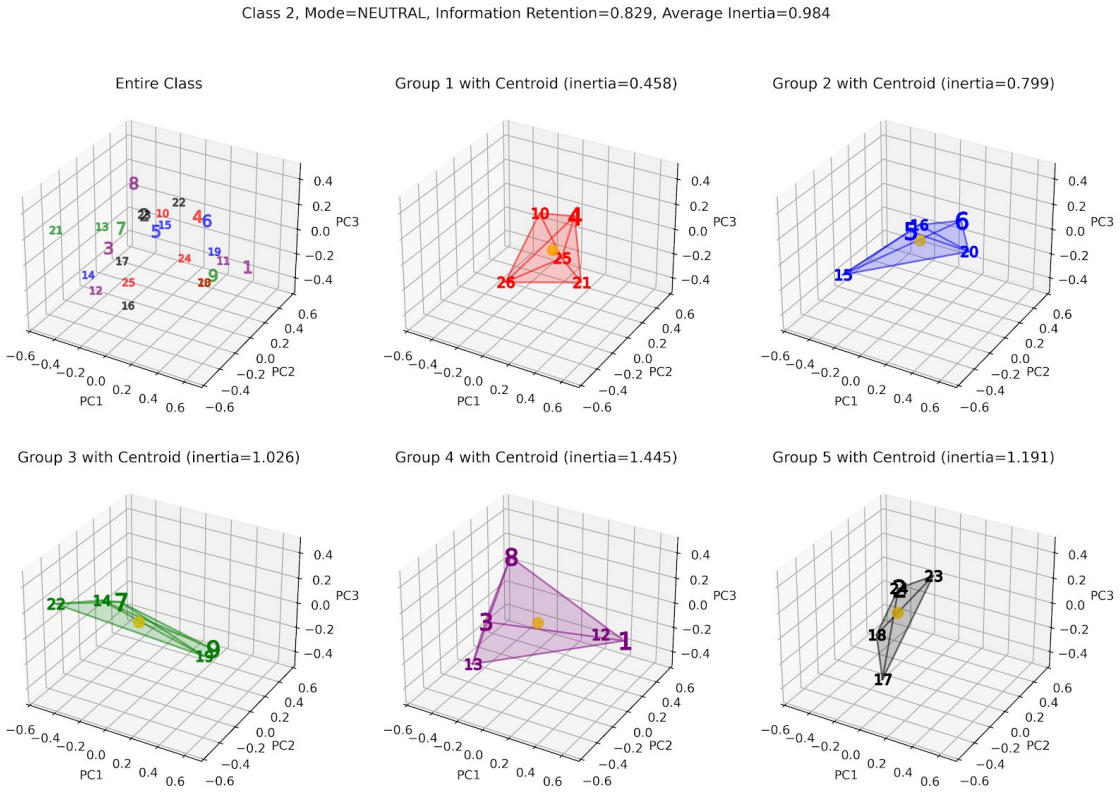

Figure 6.2.6. PCA reduced 3D visualizations of the entirety of Class 2, along with Groups 1-5 when a mix students with both similar and different character traits were grouped together.

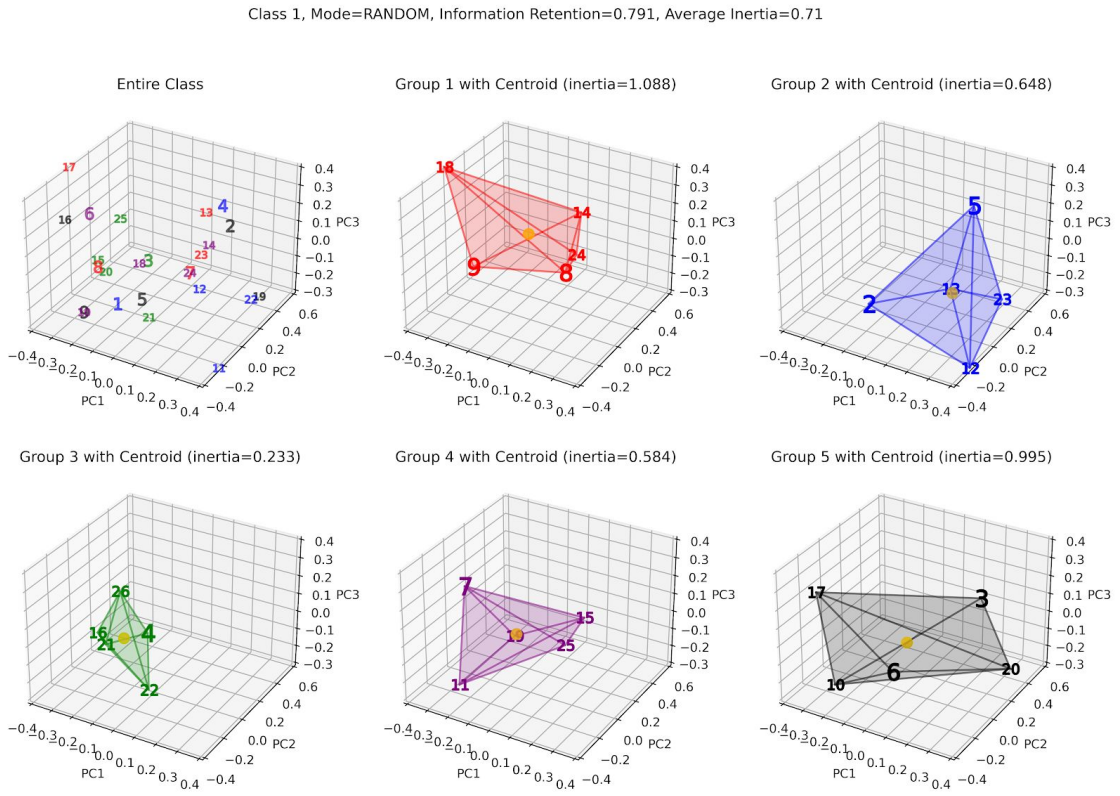

Figure 6.2.7. PCA reduced 3D visualizations of the entirety of Class 1, along with Groups 1-5 when groupings were generated randomly.

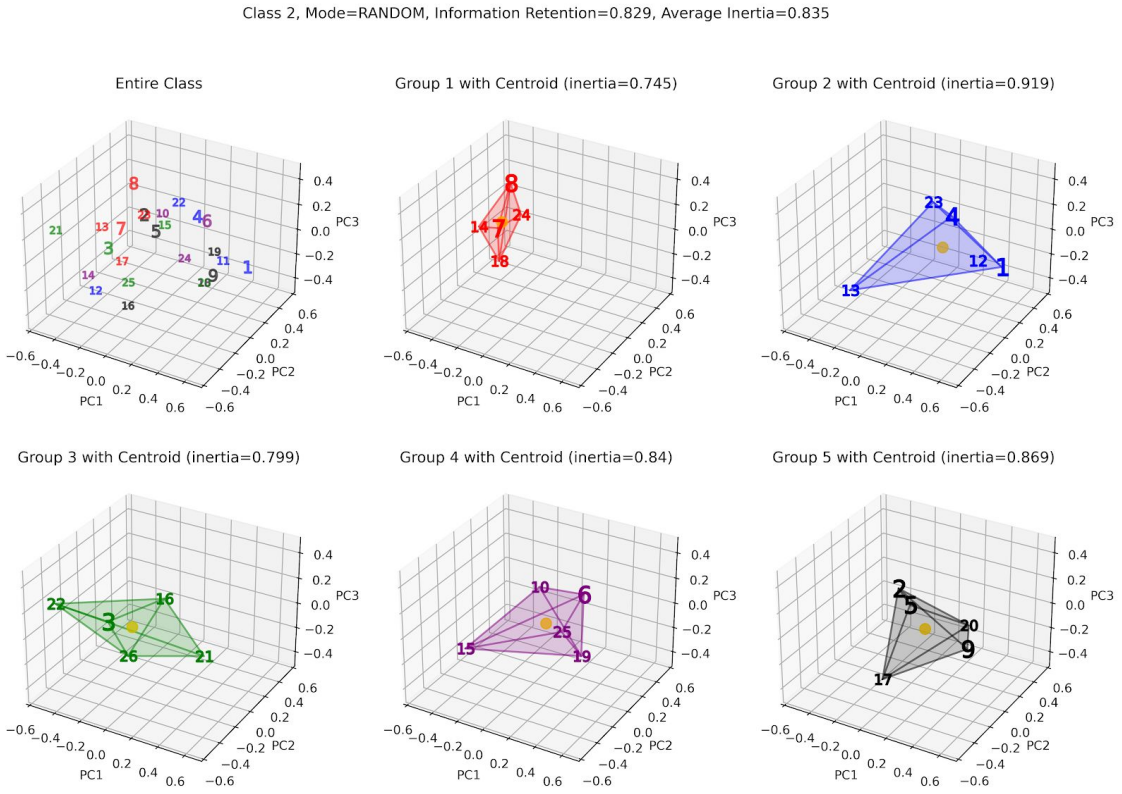

Figure 6.2.8. PCA reduced 3D visualizations of the entirety of Class 2, along with Groups 1-5 when groupings were generated randomly.

As seen from Figures 6.2.1-6.2.8, the spread of the character traits matches expectations. Where mode=DIFFERENT, the inertia of the groups is big, indicating a large spread of character strengths. Where mode=SIMILAR, the inertia of the groups is small, indicating that students of similar character strengths have indeed been clustered together. When mode=NEUTRAL and mode=RANDOM, we notice that some groups have large variation in character traits while others have small variation. As such, our genetic algorithm is indeed working in grouping students according to the three fitness functions, f_1, f_2 and f_3.

6.3. Inertia vs. Generation

Inertia is a measure of the spread in a dataset's points. It is identical to the *k-means inertia* used in the k-means unsupervised clustering algorithm. More details on how the k-means clustering algorithm works can be found in Section 9.3.

Inertia starts out from 0 and has no upper limit. However, as a rule of thumb, the lower the inertia, the more compact the dataset. In our case, this means that the group in question has students of more similar character traits. The inertia of each group in each generation's fittest configuration was calculated as per Figure 6.3.1, then the average inertia of the whole configuration was plotted against generation.

$$\text{Inertia} = \sum_{i=1}^{5} \|x_i - c\|^2$$

Figure 6.3.1. Full formula for inertia. *x_i refers to the i-th student in the group while c refers to the centroid of the group, i.e. the mean of the group's data points.*

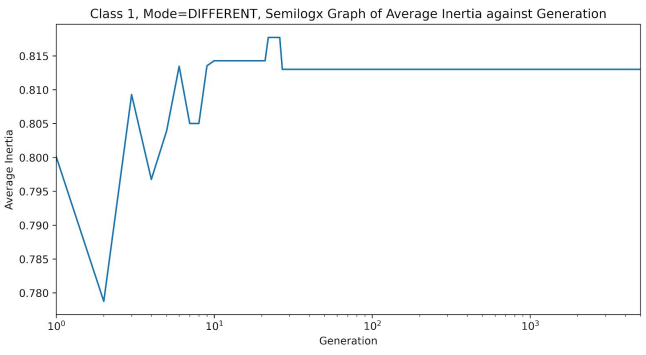

Figure 6.3.2. Semilogx Graph of Average Inertia against Generation for Class 1 when students with different character traits were grouped together.

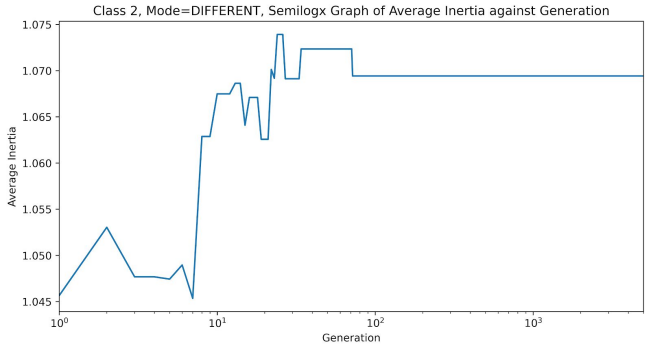

Figure 6.3.3. Semilogx Graph of Average Inertia against Generation for Class 2 when students with different character traits were grouped together.

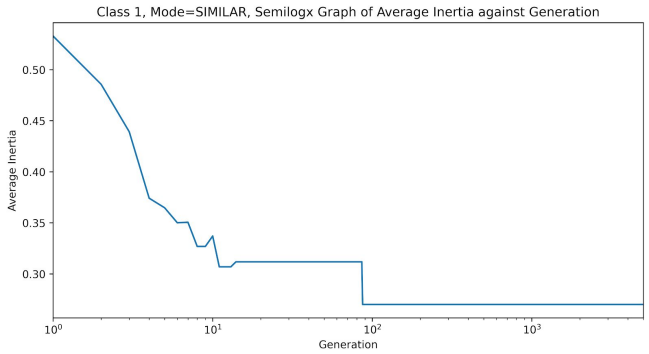

Figure 6.3.4. Semilogx Graph of Average Inertia against Generation for Class 1 when students with similar character traits were grouped together.

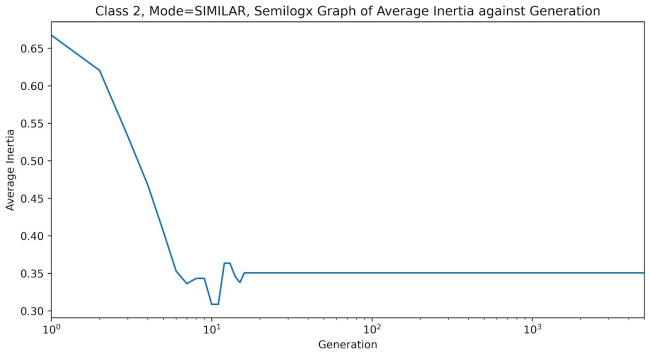

Figure 6.3.5. Semilogx Graph of Average Inertia against Generation for Class 2 when students with similar character traits were grouped together.

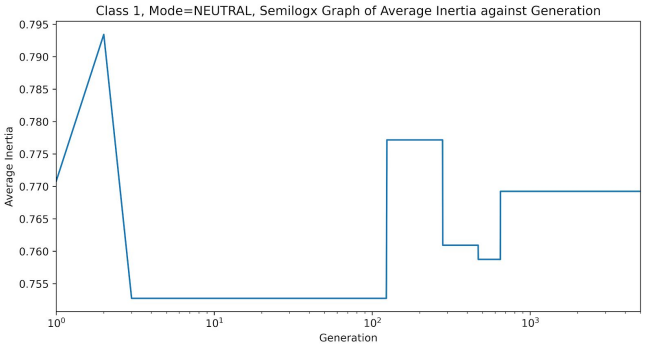

Figure 6.3.6. Semilogx Graph of Average Inertia against Generation for Class 1 when a mix of students with both similar and different character traits were grouped together.

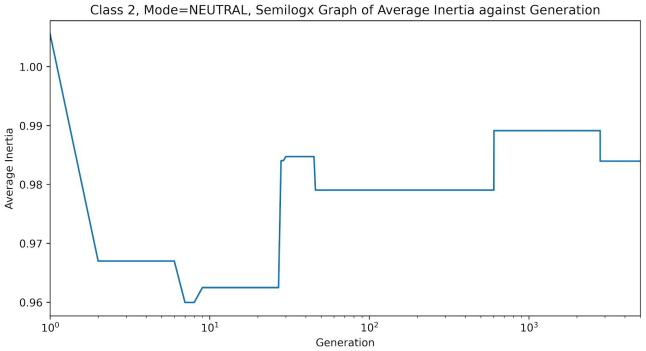

Figure 6.3.7. Semilogx Graph of Average Inertia against Generation for Class 1 when a mix of students with both similar and different character traits were grouped together.

The results from Figures 6.3.2-7 show that our genetic algorithm's performance aligns with our expectations. Where mode=DIFFERENT, the genetic algorithm aims to maximize the groups' inertia as students of contrasting character traits should be grouped together. This is reflected in the upwards trend of average inertia in Figures 6.3.2 and 6.3.3.

Where mode=SIMILAR, the genetic algorithm aims to minimize the groups' inertia as students of similar character traits should be grouped together. This is also reflected in the downwards trend of average inertia in Figures 6.3.4 and 6.3.5.

Where mode=NEUTRAL, the average inertia does not fluctuate much from its initial value. This could be because the initial random configurations generated by our genetic algorithm already possessed a decent mix of students with similar and different character strengths.

It is worth noting that for mode=DIFFERENT and mode=SIMILAR, inertia remains relatively invariant after 100 generations. This aligns with the

plateauing of configuration fitness after 100 generations, as expounded in Section 6.1.

6.4. Tukey's Range Test

To investigate if our genetic algorithm yielded groupings that were statistically significant from random groupings, we employed the pairwise Tukey's Range Test.

Given a set of datasets, the Tukey's Range Test will compare the means of every possible pair of datasets according to the null hypothesis H_0 and hypothesis H:

- H_0 = The pair of datasets have means that come from the same normally distributed population.
- H = The pair of datasets have means that do not come from the same population.

Given a significance level of α, the test will generate a p-value p, which is the probability of accepting H_0. If $p > \alpha$, H_0 is deemed valid, else H_0 is rejected and we say that the pair of datasets are statistically significant from each other.

In our case, $\alpha = 0.05$. For each class, the genetic algorithm was run 5 times to produce 5 fittest configurations for each mode (DIFFERENT, SIMILAR, NEUTRAL). Additionally, 5 random configurations were created as a control. For each top configuration of each mode, the average inertia was calculated. The Tukey's Range test was then carried out on the inertia data. The results are as per Tables 6.4.1 and 6.4.2.

Multiple Comparison of Means - Class 1			
Dataset 1	Dataset 2	p	Reject H_0?
DIFFERENT	NEUTRAL	0.0185	True
DIFFERENT	RANDOM	0.001	True
DIFFERENT	SIMILAR	0.001	True
NEUTRAL	RANDOM	0.0151	True
NEUTRAL	SIMILAR	0.001	True
RANDOM	SIMILAR	0.001	True

Table 6.4.1. Pairwise Tukey's Range Test results for Class 1.

The results from Table 6.4.1 confirm that the 3 fitness functions of our genetic algorithm generate results that are statistically significant from random groupings and from each other.

Multiple Comparison of Means - Class 1			
Dataset 1	Dataset 2	p	Reject H_0?
DIFFERENT	NEUTRAL	0.001	True
DIFFERENT	RANDOM	0.001	True
DIFFERENT	SIMILAR	0.001	True
NEUTRAL	RANDOM	**0.3926**	**False**
NEUTRAL	SIMILAR	0.001	True
RANDOM	SIMILAR	0.001	True

Table 6.4.1. Pairwise Tukey's Range Test results for Class 2.

Interestingly, the groupings generated by our algorithm when mode=NEUTRAL were statistically insignificant from random groupings for class 2. Once again, we believe that this is because random groupings could already possess a decent mix of students with similar and different character traits.

This seems to suggest that when looking for groupings with a neutral mix of similar and different traits, it could perhaps be faster to generate groupings at random.

6.5. Questionnaire Results

A total of 35 students (totalling 105 rankings) from both classes responded to the form.

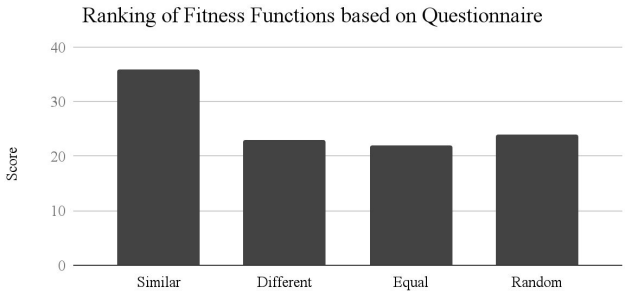

Figure 6.4.1. Graph representing the scores of each criteria. *Generating groups using the similar, different, equal, and random criteria resulted in a score of 36, 23, 22, and 24 respectively.*

The results show that grouping similar people together was the highest ranked, with a score of 36. The other groupings had scores in the range of 22 to 24. This suggests that other methods of ranking the groups do not cause a significant change in the behavior of the group.

These scores were consistent across both classes. The "similar" fitness functions were shown preference by the respondents from both classes.

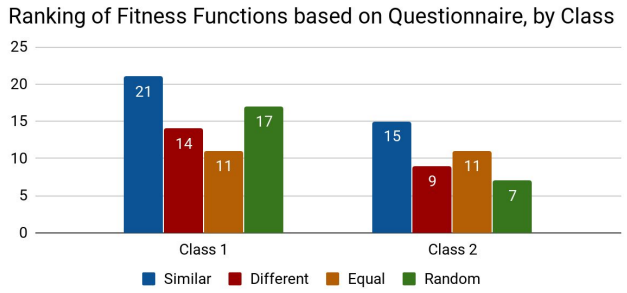

Figure 6.4.2. Graph representing the scores of each criteria, split by class. *Both classes showed similar trends, where the similar fitness function was ahead of the rest, while the other functions were clustered together, although the variance was greater.*

7. Conclusion

A genetic algorithm was successfully implemented and utilized to categorize people into groups with members of similar or different personality traits. However, where neutral mixes of similar and different personality traits were concerned, random grouping could serve as a more computationally efficient grouping method that achieves roughly the same performance as our genetic algorithm. Through a questionnaire, it was also determined that from the perspective of students, forming groups with members of similar character strengths was the most optimal.

7.1. Limitations

The genetic algorithm used in this study could be improved to accommodate varying group sizes, because the current implementation is only capable of grouping 25 people into groups of 5. Work should be done for different group sizes to be configured using this method.

Other potential factors that affect group dynamics should be considered in the “gene pool” as well. These include, for example, the individuals' interests, abilities, gender and age[6], as the top 5 VIA character strengths certainly have their limitations in painting a complete picture of an individual.

Also, the number of respondents for the form was relatively low (35), resulting in a larger probability for the results to have been skewed due to subjective reasons. Additionally, the respondents already had long periods of time to bond with each other before this study, leading to added potential bias, as respondents may rank those who are closer to them higher. However, this could also have been beneficial to group dynamics, due to their better understanding of each other. A study into how the results were affected by this has not been conducted.

A wider range of classes should also have been considered. Both engaged classes were from the accelerated Science programme, and this may result in the results being valid only for classes with a similar profile. Classes from Arts streams and from other educational pathways should also be considered for a more comprehensive understanding of whom these results apply to.

The results from the questionnaire were not tested in real life, and relied completely on the personal opinions of respondents about their groupings. An objective analysis of the performance of each grouping was not conducted.

7.2. Future work

Fitness functions utilizing high-dimensional distance metrics other than the cosine similarity could be tested. As the code was implemented in a modular fashion, it is possible and comparatively simple to swap the fitness function out, and achieve different results. The best algorithm could then be found in a manner similar to the questionnaire.

Another questionnaire should also be performed, but with a larger set of respondents. Additionally, a real-life performance evaluation of each group can be conducted, in order to minimize the subjectiveness of the results. This can be achieved by, for example, having each assigned grouping perform the same task, and monitoring the number of arguments that break out, the efficiency of every group, or the quality of the products produced as grading factors. This would allow for a more rigorous evaluation of whether our genetic algorithm truly engendered improved group dynamics.

8. Bibliography

1. The 24 Character Strengths. (2020). Retrieved May 25, 2020, from https://www.viacharacter.org/character-strengths

2. Prabhakaran, S. (2020, April 28). Cosine Similarity - Understanding the math and how it works. Retrieved May 25, 2020, from https://www.machinelearningplus.com/nlp/cosine-similarity/

3. Cosine similarity. (2020, May 17). Retrieved May 25, 2020, from https://en.wikipedia.org/wiki/Cosine_similarity

4. Mallawaarachchi, V. (2020, March 01). Introduction to Genetic Algorithms. Retrieved March 1, 2020, from https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3

5. (somewhere on geeksforgeeks)

6. Yang, Q., & Chen, L. (2013). A learning grouping algorithm based on user personality. 2013 8th International Conference on Computer Science & Education, 71-75. doi:10.1109/iccse.2013.6553886

9. Appendices

9.1. Source Code

The source code for this project can be found at https://github.com/jzh001/Genetic-Algorithm-for-Group-Classification.

9.2. Detailed explanation of how PCA reduction works for our case

To perform PCA reduction, the data was first normalized to the range of 0-1. Subsequently, a *covariance matrix* was calculated to investigate the relationships between each variable in the dataset. This was done by looking at the *covariance* between every possible pair of datasets, which is a measure of the variance of data points from the mean of each dataset. Covariance can be calculated as per Figure 9.2.1:

$$\mathrm{Cov}(x,y) = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{n}$$

Figure 9.2.1. Full equation for covariance. *x_i is the i-th data point from dataset x, while $\bar{x}$ refers to the mean of dataset x. The same goes for y, y_i and $\bar{y}$. n refers to the number of data points in either dataset x or dataset y, as they are supposed to be equal in length. If $Cov(x,y) > 0$, there is a direct relationship between variables x and y. If $Cov(x,y) < 0$, there is an inverse relationship between the 2 variables.*

For our 6D character strength vectors with 6 variables, the covariance matrix is per Figure 9.2.2:

$$\begin{bmatrix} \mathrm{Cov}(S_1,S_1) & \mathrm{Cov}(S_1,S_2) & \cdots & \mathrm{Cov}(S_1,S_5) & \mathrm{Cov}(S_1,S_6) \\ \mathrm{Cov}(S_2,S_1) & \mathrm{Cov}(S_2,S_2) & \cdots & \mathrm{Cov}(S_2,S_5) & \mathrm{Cov}(S_2,S_6) \\ \vdots & & \ddots & & \vdots \\ \mathrm{Cov}(S_5,S_1) & \mathrm{Cov}(S_5,S_2) & \cdots & \mathrm{Cov}(S_5,S_5) & \mathrm{Cov}(S_5,S_6) \\ \mathrm{Cov}(S_6,S_1) & \mathrm{Cov}(S_6,S_2) & \cdots & \mathrm{Cov}(S_6,S_5) & \mathrm{Cov}(S_6,S_6) \end{bmatrix}$$

Figure 9.2.2. Covariance matrix for 6D PCA reduction. *In this matrix, S_i $(1 \le i \le 6)$ refers to the i-th broad category of VIA character strengths (see Section 5.1).*

Next, the 6 principal components were computed by performing an *eigendecomposition* of the covariance matrix. The eigendecomposition seeks to transform the 6x6 covariance matrix into 6 *eigenvectors* and 6 *eigenvalues*. Each eigenvector is associated with its own eigenvalue, and together they constitute the vector equation of a 6D line. This vector equation is synonymous with a principal component, and can be represented as per Figure 9.2.3:

$$l : \vec{r} = \vec{O} + \lambda\vec{E}$$

Figure 9.2.3. Vector equation of the principal component (6D line in our case). *$\vec{O}$ refers to the origin. λ is the eigenvalue that represents the variance explained by $\vec{E}$, the eigenvector or direction vector of this 6D line.*

As aforementioned, the first principal component seeks to retain the majority of the information in the original dataset. As such, the first principal component's 6D line aims to maximize the variance of the dataset, thus explaining the most deviations in the data points. The second 6D line is constructed such that it also maximizes the variance of the dataset, but is orthogonal to the first

line, thus explaining its decorrelation with the first line. The third, fourth, fifth and sixth principal components are calculated in the same way.

The 3 most important principal components were then selected and placed into a feature vector as per Figure 9.2.4. The feature vector is named so because it only contains the relevant features, or principal components that are of interest so as to do the actual dimensionality reduction.

$$\begin{bmatrix} \vec{E}_1 & \vec{E}_2 & \vec{E}_3 \\ \lambda_1 & \lambda_2 & \lambda_3 \end{bmatrix}$$

Figure 9.2.4. 2x3 feature vector containing the 3 most important principal components. *Once again,* λ_i *is the i-th eigenvalue, and* $\vec{E}_i$ *is the i-th eigenvector (*$1 \leq i \leq 3$*).*

Lastly, to rescale the original dataset along the axes of the 3 principal components, the transposition of the feature vector was multiplied by the transposition of the original dataset.

9.3 Detailed explanation of the mechanism of the k-means clustering algorithm

The k-means clustering algorithm seeks to cluster a set of N data points into K clusters. It works through the following steps:

1. Place k random centroids of the initial clusters in the same data space as the N data points. Take note that these centroids are not part of the dataset.
2. Using a suitable distance metric, calculate the distance from every data point to every centroid, and assign the data point to the closest centroid.
3. Based on the above assigned data points in step 2, recompute the position of the centroids by taking the mean of the data points.
4. Repeat steps 2-3 until the centroids move by negligible amounts, after which the k-means algorithm has deemed to have converged to a solution to the clustering problem.