Department of Computer Science

Submitted in part fulfilment for the degree of BEng

Group formation tool for collaborative learning

Yuqing Gong

23 June 2021

Supervisor: Nicholas Matragkas

Acknowledgements

I would like to thank Nicholas Matragkas for all their guidance and advice throughout this project.

Statement of Ethics

This project raises no ethical concerns as it simply analyses anonymous data. All data used in this paper were self-generated and thus may not represent real world applications.
The result of the website will be deleted immediately by the server.

TABLE OF CONTENTS

TABLE OF FIGURES

Executive summary

Collaborative learning has been widely used across different disciplines in the process of teaching. It involves groups of students working together toward a common goal. Group report and group presentation are examples of collaborative work which has been adopted as an effective educational approach to enhance the learning process and the overall learning experience.

It is claimed by researchers that many unsuccessful outcomes of collaborative work are from the process of group formation since several student characteristics have to be considered simultaneously. While substantial research is being conducted into the advantages of collaborative learning, there is only little attention on how to create efficient groups for students. Moreover, there is little open source computer-based tool to assist the process of group formation.

The group formation problem has become popular in recent years. In this problem, students that match a required set of skills must be chosen as a group in order to maximise the group performance.

In order to solve the group formation problem, this project has two main objectives. The first objective is to compare genetic algorithms and clustering-based algorithms, which are two common ways to form groups. As literature suggested, heterogeneous groups work better in most of the group assessments. Therefore, in our solution, the genetic algorithm runs until a best solution, i.e., the most intra-heterogeneous and inter-homogeneous result, is found within limited time, where inside groups, the heterogeneity of students remains maximum in order to achieve ability diversity, but the difference among groups should be as small as possible to keep fairness for all groups. The clustering-based technique computes the similarities of students by calculating the distance among their characteristics and then allocates similar students to different groups until all students are assigned. The ability of these two algorithms will then be compared by analysing their performance and scalability after obtaining the standard deviations and execution time of the results that are gained from the implementation of the algorithms using a set of randomly generated students' data.

The second objective is to create an extensible architecture for a web tool that implements these algorithms in order to address the lack of

tools. Via Heroku, a cloud platform, this web tool is able to provide some basic functions. For example, users can upload students' data in the form of csv file to the tool, create students' groups based on different criteria and download the solution of group composition from the tool.

The result is that both genetic algorithm and clustering algorithm have stable and high standard deviations which shows that the proposed methods are able to create heterogeneous groups while assigning students to teams.

The results were obtained from a series of self-generated students' data and therefore may not represent real-world applications. Testing on the algorithms may need to be done on real students' data and have team members' academic scores recorded after forming the teams in order to verify the performance of the groups. More work could be done in determining the heuristics of homogeneous groups so that the educator can choose the group type by themselves.

There are little social or legal issues here as students' data used in this report are generated randomly and anonymous.

1 Introduction

Collaborative learning (CL) is pointed out as an important pedagogical tool for group assessments and has been suggested to have a positive impact on students [1]. While some groups have high marks, others may only have low achievement. Research has shown that many of the unsuccessful learning outcomes are from the process of group formation [4]. Therefore, the formation of groups is a fundamental prerequisite which will be the focus of this report.

In the CL process, one of the major challenges faced by educators is how to form groups in a way that students can interact with each other more effectively. This task is challenging because while splitting a cohort of students into different groups, a lot of factors need to be considered. For example, how many students should be assigned to each group? How to select suitable teammates for students? What criteria need to be considered in order to improve interaction within a group? What skills are needed for a group to fulfil the tasks successfully? How to measure students' difference if the characteristic is besides grades such as gender and ethnicity? The process of group formation is especially difficult when considering a large number of students due to the diversity in students' characteristics and large number of combination possibilities [2].

Moreover, although there is plenty of prior work in the area of Collaborative Learning, describing the advantages and analysing how teams can best be made, there is no open source online tool to automate the process of assigning students to collaborative groups. For big cohorts, it could be hard and inefficient to assign students to groups just by hand. There are a few ways to form groups including metaheuristics, constraint programming, clustering, etc. In this report I have implemented a flexible architecture which enables the comparison of these approaches and I have implemented algorithms from two of these families of techniques to test the architecture. Therefore, this report has the following two objectives:

1) To use a genetic algorithm and a clustering algorithm to form student groups and then assess the advantages and disadvantages of the algorithms.
2) To develop a tool for group formation which is performant and extensible for the algorithms.

Some studies propose strategies with the main goal of obtaining heterogeneity inside the group, insisting on the hypothesis that individual differences inside a group encourage better meaningful interaction and students who have better results can help their groupmates [5][12]. There are various methods to form heterogeneous groups. One approach that has been suggested is to use a genetic algorithm. For example, Ani et al [2] use GA to form groups based on students' programming skill so that every group will consist of students who are good, moderate and poor at programming. However, the algorithm of fitness calculation only focused on one particular characteristic which may not fit a real world situation.

To solve this problem, our report first introduces a genetic algorithm considering an **arbitrary number of criteria** of students. The GA will produce the best individual for each generation and output the most optimal solution within the maximum generation time, where inside each group, the diversity of students is as large as possible, and among groups, the difference will remain minimal to keep the fairness and balance for all groups. The diversity and the homogeneity of each feasible solution is measured by a fitness score which is a series of iterative calculation of the mean square difference between the whole sample and each group. The implementation is based on previous work conducted by Julián et al [3] but extended to allow it to be run on a web. We also use a clustering-based algorithm to solve the above problem. The work first looks at students' characteristic difference and picks k most similar students to k groups randomly each time until all students are assigned. The implementation is a modification of [6] so that a more effective group formation could be created. This requires a way to calculate the centroid of students' data which will be present in this report.

Our web-based tool is developed to assign students to groups using pre-defined criteria. First, the instructor chooses which characteristics of students to measure in creating groups. Next, depending on the criteria, the instructor may need to obtain the data from student surveys or databases. Finally, the instructor uploads the data to the tool and the system will assign students to teams automatically and the result can be download. The web-based tool provides a web interface with instruction clearly stated and it forms student groups in a more effective way.

The rest of the report is organized as follows: In Session 2, literature is reviewed briefly, describing the existing grouping strategies and grouping formation algorithms. Session 3 describes the problems that need to be solved in this report. Session 4 presents the proposed methods and architecture for the problems focusing on algorithmic formulation and Web interface, whereas Section 5 presents the experiments with randomly generated students' data, shows the results and then evaluates the performance and scalability of the algorithms. Finally, Section 6 presents the concluding remarks and some directions for future work.

2 Literature Review

This section will review research around the topic of group formation in Collaborative Learning. This is particularly helpful for our project if we know what has already been done in terms of group design and implement method, which are key factors of group formation problem. Therefore, the project began by going through the relevant papers, ensuring we understand our subject matters and the methodologies that are currently available for them.

2.1 Grouping Strategy

In this session we will examine work on the aspects of grouping type, grouping criteria and grouping size, namely, the grouping strategies.

2.1.1 Grouping Type

At the start of a group formation process, one important factor that needs to be considered is the type of group we want to have. Most of the literature focuses on one of the two types: Homogeneous group or heterogeneous group [7].

Quite a number of researchers pointed out the positive effect of the diversity on a group, insisting the difference in student’s abilities and attitudes are more likely to generate meaningful interaction among students and increase the performance of the groups [2], [3], [4], [5], [6], [8], [10], [15]. It has been observed by Gogoulou et al [13] that heterogeneous groups out-perform homogeneous groups in a wide range of tasks. Pang et al [6] stated that the diversity of students will increase the knowledge pool and lead to a positive effect on cognitive complexity. Also it is suggested by Razmerita and Brun [8] that heterogeneous groups tend to be more innovative and creative, which is especially effective for individual learning. However, Lambić et al [15] pointed out that high-ability learners in heterogeneous groups are more likely to be dissatisfied as it requires additional time and effort to help low-ability learners, on the other hand, low-ability learners may feel left-behind and uncomfortable as they need help from other peers. Therefore, it is suggested that although students work best when groups contain a combination of high- and low-ability learners, great differences among group members can decrease the effectiveness of cooperation.

The other type of group is homogeneous group where students in the same group have similar ability, learning style or topic interest. Lambić et al [15] stated that this kind of group often has high sociability and high cohesiveness. Also in an all-high ability homogeneous group, learners can communicate and learn at a faster speed due to the same level of understanding. However, Lambić et al [15] also pointed out that there is a lack of multiple perspectives in all-low ability homogeneous groups and learners may have insufficient opportunities for improvement as they tend to be more hesitant to ask questions or ask for help.

Christodoulopoulos and Papanikolaou [9] and Dlab and Bozic [12] stated that both group types are helpful in achieving specific goals depending on several factors such as type of tasks, student's abilities and curriculum area. Razmerita and Brun [8] stated that while heterogeneous groups are better for long-term knowledge discovery tasks, homogeneous groups are better for short-term and specific guided problems.

2.1.2 Grouping Criteria

Another factor that needs to be considered is grouping criteria. Collaborative Learning has many variables and various factors could impact the group performance. Therefore, an instructor may define a set of student characteristics or constraints to measure in splitting the cohort of students into different groups. A large number of researchers use Knowledge levels/ Skills/ Academic performance as criteria to form the groups in order to minimize learners' abilities imbalance [1], [2], [3], [4], [6], [8], [9], [10], [11], [12], [13], [14], [15]. In order to achieve demographic diversity, several researchers have taken Ethnic background as their grouping criteria [1], [6], [8], [10]. Previous research [1], [3], [5], [6], [10] and [12] took the biographical attribute, gender, as one of the student characteristics, in addition to this, Moreno et al [3] and Pang et al [6] also group students based on learners' ages. Studies in [1], [3], [5], [9], [12], [13] and [14] showed that the performance of a group could be influenced by students' learning styles/ cognitive styles, where Wang and Kojima [5] and Christodoulopoulos and Papanikolaou [9] both suggested to use the levels of sensing and active learning styles to assign learners. Constraints based on personality traits have also been proposed by various researchers, those personality attributes include Shyness [3], [4], Self-confidence [4], Achievement motivation [3], [4], [5], Group work attitude/ Leadership [3], [4], [6], [13] and Language fluency [4], [6]. Topics of interest/ Project preference as a grouping constraint have been selected [4], [6], [8], [11], [12], [20]. Study program is also a grouping constraint which was used by Smith and Spindle [1], Razmerita and Brun [8] and Dlab and Bozic [12]. In addition, it is stated that different weights could be assigned to different criteria due to various importance [6], [9], [11], [12], [14], [22].

2.1.3 Grouping Size

When forming the groups for students, the group size needs to be taken into account by the instructor. It is stated that Cooperative Learning (CL) is based on small group processes and learners should be divided into small teams [1], [5], [8], [9], [10], [12], [14], [21]. It has been recommended by Graf and Bekele [4] and Lambić et al [15] that learners should work in groups of **four** members where each of the group consists of one high achiever, one low achiever and two average achievers. It is determined that the group size of three to five [10], [12], [13], [22] whereas Lin et al [20] reported the optimal group size to be five to eight.

2.2 Group Formation Techniques

Recently, a wide variety of computer-based methods have been used in group formation [15]. In this session, we will review the designed grouping algorithms which have been proposed in previous studies. Those algorithms include metaheuristic algorithms and clustering algorithms.

2.2.1 Metaheuristic Optimization Algorithms

Metaheuristics are higher-level heuristic approaches [17]. They refer to stochastic optimization, which contains a wide range of algorithms and techniques that find the near optimal solution of a sophisticated problem. It has been observed that most metaheuristics are elaborate combinations of Hill-climbing and Random search [16]. Metaheuristics maximize the outcome with limited budget and computational time by selecting the best solution from a set of available solutions and the final solution, namely, the best solution discovered, is returned when a converged state has been reached [17]. Although there is no guarantee that the result found will be globally optimal or high quality [19], metaheuristic algorithms tend to provide satisfactory solutions at an acceptable computational effort and solve the problems quickly [3].

Therefore, a wide range of studies adopted the metaheuristic algorithms in their group formation processes. Next, we will review paper that used metaheuristic algorithms for group formation, including Genetic Algorithm, Ant colony optimization, Particle Swarm Optimization and Hill Climbing Algorithm.

2.2.1.1 Genetic Algorithm

Ani et al [2], Moreno et al [3], Imbrie et al [10] and Gogoulou et al [13] all used genetic algorithms to form their student groups. Genetic algorithms take ideas from Darwinian Natural Selection [3]. They encode the potential solutions of the problem by generating a discrete integer-based chromosome structure and a satisfactory solution will be found after a series of iterations of recombination operators [2], [10].

The GA begins with an initial population represented by N individuals randomly. Moreno et al [3] stated that an individual is a feasible grouping composition and is represented as a chromosome composed of IDs where each ID makes reference to a student. Then each individual's quality will be measured using a certain fitness function. Later, the reproduction process will be applied based on the fitness values. A common method for the selection process is Roulette-wheel selection, a solution with higher fitness score is more likely to be selected. This method has been used by Ani et al [2], Moreno et al [3], Imbrie et al [10] and Gogoulou et al [13]. Then the standard GA operators, i.e., crossover and mutation, will be applied to the selected solutions to generate new better individuals. Those new individuals will represent another new population, which is the next generation or offspring [2]. This process is repeated until a stopping criterion is met, i.e., reach the maximum iteration times, and then the current best solution will be the result. A best solution indicates the most inter-homogeneous and intra-heterogeneous group composition [2], [3], [13].

One advantage of the GA is that it is a popular and well-established approach which could be used for many applications. After running the experiments, Moreno et al [3] stated that the proposed GA is able to obtain inter-homogeneous and intra-heterogeneous groups and it generates better solutions in terms of grades than other methods such as self-organization or assigning randomly. Imbrie et al [10] observed that the GA can automatically assign a large number of students to teams in less than one hour. It could converge to a promising result with high quality easily and keep the optimal fitness over several generations. Moreover, with only little adjustments, this method could be used in different situations and programming platforms.

However, Moreno et al [3] also pointed out the limitation of GA is that the instruction must consider the incorporation of particular criteria carefully, and it is best to use an exhaustive method when two student characteristics are considered and the number of students is low. Also it requires a large number of iterations and high implementation cost due to large GA operators. What’s more, the random heuristics might not always find the optimum.

2.2.1.2 Ant Colony Optimization

Graf and Bekele [4] proposed the ACO (Ant colony optimization) algorithms inspired by the collective foraging behaviour of specific ant species. These ants will drop a chemical substance called pheromone on its chosen path while looking for food. Then other ants can therefore sense the environment and tend to follow the path that has more pheromones on. If a path is selected frequently, its pheromones will increase thus increasing the probability that an ant will choose it again. It is observed that an ant may not always follow the path that has the highest pheromones since the paths with less pheromones could also be chosen with a low probability. However, pheromones evaporate over time, and the rarely used trails will vanish which indicate the best paths for ants to find the food.

To form the groups with ACO, Graf and Bekele [4] ordered students comparable to the travelling salesman problem, where students in the same group are linked to each other and the decision which student starts a new group is performed randomly. By measuring the local information of an edge through calculating the sum of the Euclidean distance of the new student and all already assigned group members, we could know the benefit of adding the newly assigned student to an existing not-full group. Later, the global information is calculated in order to update the pheromones on the edges between the new student and all other group members. Figure 1 is a graphical representation of the algorithm.

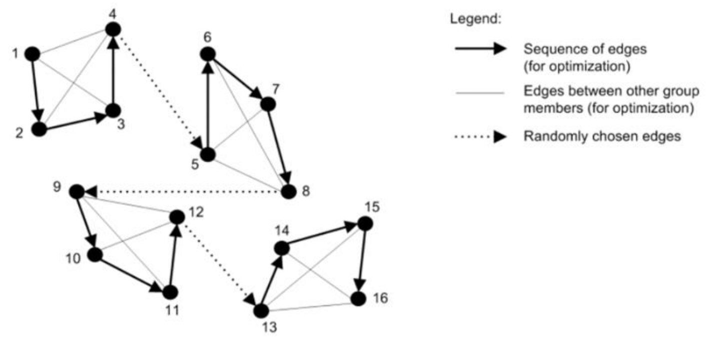

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
Legend:
Sequence of edges (for optimization)
Edges between other group members (for optimization)
Randomly chosen edges

Figure 1. Representation of the grouping problem as graph (group size = 4) [4]

Graf and Bekele [4] observed that the ACO is capable of searching near-optimal solutions among a population in parallel under different student datasets. It also can adapt to new distances and have guaranteed convergence. Nevertheless, the time of convergence is not certain and the theoretical analysis is relative difficult.

2.2.1.3 Particle Swarm Optimization

Lin et al [20] proposed a Particle Swarm Optimization algorithm which is inspired by swarm intelligence of fish or birds. While searching for food sources, those species often work collaboratively. For example, a group of birds are flying randomly to search for one piece of food. They don’t know where the food is, but they know how far they are in each iteration, and they will follow the bird which is the nearest to the food to drive to an optimal path.

PSO involves the use of a population of candidate solutions, namely, a swarm of particles; those particles distributed over a large space are potential solutions to the problem and PSO is able to improve the candidate solutions by updating generation. Each particle keeps track of its locations and moves towards the best known position in the N-dimensional problem space by using a fitness function and a velocity function.

For our group formation problem, PSO begins with a population which consists of some random grouping solutions and then each solution searches for the optimum value by updating its velocity for its next generation. Lin et al [20] also introduced the Genetic Algorithm to ensure the best particle in the updating process services.

Lin et al [20] stated that EPSO is efficient in obtaining the global optima and thus saving time especially when assigning a large number of students. The robustness of EPSO has been proved under multiple experimental conditions. What's more, it has a low number of iterations and it only needs to adjust very few parameters. However, its performance is influence by the initial random generated numbers. And it doesn't guarantee the result is the best solution.

2.2.1.4 Hill Climbing Algorithm

Cavanaugh et al [22] explored the use of hill-climbing algorithm to assign students to groups. To form the groups, the algorithm begins with a specific team size which is defined by the instructor and then students will be assigned randomly into a non-full group until all are assigned. Later, the heuristics and two fitness scores, i.e., before and after swapping students between two groups, will be evaluated. The algorithm will keep the set of teams with higher fitness scores and repeat the swapping until it finishes. This process is repeated many times in order to find the global maximum and an optimal set of groups at a low execution time.

Hill climbing algorithm is relatively easy to implement to solve the group formation problem at a low computational cost as it doesn't store a search tree. However, the initial solution will affect the optima. and it might get stuck in local maxima.

2.2.2 Clustering Algorithms

Christodoulopoulos and Papanikolaou [9] stated that some meta-heuristic algorithm's computational complexity is high and implementation is quite hard for purely web-based environments that use JavaScript and PHP, therefore clustering algorithms are alternative methods to automatically form student groups [8], which are a category of optimization algorithms for grouping data [14]. The main traditional clustering algorithms include k-means, hierarchical clustering algorithms, fuzzy c-means and model-based clustering approaches [25]. The number of clusters, the size of the cluster and the quality criteria are defined depending on its chosen algorithm. According to Razmerita and Brun [8], one widely used method to measure the distance, i.e., the similarity between students in groups, is the Euclidean distance [Figure 2], then the instructor may assign students from clusters to groups based on the type of groups he/she wants to create. In this session, we will review the literature that proposed a C-means clustering algorithm and a novel clustering algorithm.

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2}.$$

Figure 2. The formula of Euclidean distance [29]

2.2.2.1 Fuzzy C-Means Clustering

Christodoulopoulos and Papanikolaou [9] used Fuzzy C-Means clustering (FCM) algorithms to address the group formation problem. FCM creates a membership matrix (U) for each cluster to represent the membership probabilities for each data point and thus group members can be swapped among groups based on the probability of belonging to those groups. The FCM begins with initializing probability matrix with random values and then the cluster centre of each cluster will be calculated. Later, the distance matrix for every dimension will be calculated to generate a new probability matrix and an objective function will be used to regroup the data points.

Christodoulopoulos and Papanikolaou [9] stated that the proposed Fuzzy C-Means clustering algorithm has low complexity and it is able to converge to produce heterogeneous and homogeneous groups with good quality. Nevertheless, the FCM has long computation time and it is very sensitive to the initial parameters and noise.

2.2.2.2 Clustering-based Algorithm

Pang et al [6] and Gogoulou et al [13] both employed a clustering-based algorithm. According to Pang et al [6], clustering algorithm is an unsupervised learning algorithm which seeks out similarity between pieces of data to determine whether they can be characterised to the same cluster, for example, K-means algorithm begins with defining the number of cluster (k) and the centroid of each cluster. Then the distance of each data point to those centroids will be calculated to determine which cluster they should belong to. Later the centroid of each cluster will be updated according to their cluster means. This process is repeated until termination conditions are met.

Pang et al [6] proposed a novel clustering-based algorithm, firstly it assigns students to size-balanced clusters, where inside each cluster, students have similar characteristics. Then the algorithm will pick one student from each cluster to new collaborative groups to achieve diversity.

Gogoulou et al [13] observed from the experiment results that the performance of the proposed clustering algorithm and the proposed GA are almost the same in case of heterogeneity. The clustering algorithm is very simple and it is able to produce a satisfied solution with very few programming. However, clustering algorithms cannot handle noisy data and outliers.

2.3 Conclusion

Literature suggested that there are three types of student groups: Heterogeneous groups, homogeneous groups and mixed groups. The main algorithms to automate the process of group formation include meta-heuristic algorithms such as Genetic algorithm, Ant colony optimization algorithm, Particle swarm optimization algorithm and Hill climbing algorithm, and clustering algorithms such as Fuzzy C-means clustering algorithm and a novel clustering-based algorithm. These algorithms are able to assign students to collaborative groups based on a set of student characteristics. The advantage of GA is that it could converge to a promising result with high quality easily and keep the optimal fitness over several generations. However, it requires a large number of iterations and high implementation cost due to large GA operators. The clustering algorithm is very simple and it is able to produce a satisfied solution with very few programming. However, clustering algorithms cannot handle noisy data and outliers.

Grouping criteria can include academic performance, ethnic background, gender, age, learning style, personality traits, language fluency and topic of interest. In order to produce a better result, the weight for different criteria has also been suggested by prior work. It is recommended that the group size ranges from 3 to 8.

3 Problem Description

3.1 Main Goals

Many algorithms have been proposed in the area of Collaborative Learning to solve the group formation problem, while metaheuristic and clustering algorithms are the predominant methods. Therefore, this report will present a genetic algorithm and a clustering-based algorithm and then compare their efficiency in terms of their performance and scalability. The implementation of two algorithms leaves the educators the opportunity to choose which algorithm they want to use to create their student groups.

Prior work also suggested that in most of the collaborative tasks, heterogeneous groups work better than homogeneous groups. Given by the benefits of the diversity of a group, this report will focus on the creation of intra-heterogeneous and inter-homogeneous groups, where inside each group, the heterogeneity of students remains maximum in order to achieve ability diversity, but among groups, the difference will be as small as possible to keep fairness.

Since a lot of literature only considered a limited number of student characteristics, our report will consider an arbitrary number of student characteristics to increase the flexibility for the real application. This project will develop a web-based tool which is performant and extensible to support the automated process of group formation since there is no open-source tool using arbitrary criteria to automatically optimize group formation of students.

3.2 Requirements

Single Statement of Need (SSON): The project aims to create an extensible web-based tool for group formation which involves the use of a metaheuristic algorithm and a clustering-based algorithm and then compare the efficiency of the algorithms in terms of their performance and scalability for this problem.

The project's requirements were formed from detailed project specifications and discussions with a project supervisor, who is also the customer for this project, to allow any confusion to be quickly resolved in order to ensure that the final product is as close to the stakeholders' expectations as possible. The requirements are presented by adapting the concepts shown in the IEEE Requirements

Specification [28]. This allows them to be clearly organised and referenced later on. This table-based method also allows them to be laid out in a manner that is easy to follow.

The most important part of the project is ensuring that all stakeholders are satisfied with the final result. The stakeholders are project supervisor and the web-based tool end users. Each of them has a different interest in the project and different requirements of the web-based tool. Throughout the process the needs of each of the stakeholders have been considered closely when eliciting each of the requirements. By considering the needs of the stakeholders at an early stage, it allows to identify what should be prioritised in the development of the web-based tool and what aspects are needed or less important.

We have categorised the requirement into three types: User Requirements [Figure 3], Functional Requirements [Figure 4] and Non-Functional Requirements [Figure 5]. User Requirements are those the user will find important such as creating groups; Functional Requirements are those that form part of the core element of the web-based tool with Non Functional Requirements being those that are not entirely necessary but will improve the system for the end user. Priority is given to each user requirement classing these as shall, should or may. These priorities were based on how important it is to the stakeholders for that given feature to be implemented during the development of the web-based tool. This allows better focused when delegating time in the development process, resulting in a more efficient and elegant solution. Below are the set of user requirements and how our solutions which are functional and non-functional requirements meet the user requirements. They will be used to design the system.

User Requirements

Requirement ID	Requirement detail	Priority
UR_GROUP_FORMATION	The user shall be able to use the tool to assign students to groups using arbitrary student characteristics.	shall

UR_CHOOSE_ALGORITHM	The user shall be able to choose which algorithm is going to be used.	shall
UR_SAFETY	The user shall be aware of how their data will be handled and feel confident about the safety of their personal information while using the tool.	shall
UR_GAIN_RESULT	The user shall be able to get the result.	shall
UR_DIFFICULTY	The tool should be easy to use.	should

Figure 3. User Requirements

<u>Functional Requirements</u>

Requirement ID	Requirement detail	User Requirement
FR_UPLOAD_DATA	The tool must allow users to upload a csv file that includes students' names and their characteristics.	UR_GROUP_FORMATION
FR_INPUT_GROUP_NUMBER	The tool must allow users to input the group number students need to be assigned to.	UR_GROUP_FORMATION
FR_INPUT_TYPE	Inputs are validated to only allow applicable values.	UR_GROUP_FORMATION
FR_CSV_DATA	Students' information must be written in the correct format.	UR_GROUP_FORMATION
FR_CHOOSE_ALGORITHM	The tool must allow users to choose an	UR_CHOOSE_ALGORITHM

	algorithm they want to use.	
FR_INPUT_REQUIRED	All the fields are required in order to go to the next step.	UR_GROUP_FORMATION
FR_IMPLEMENTATION	The tool must be able to process the uploaded csv file and algorithm at backend to find a solution.	UR_GROUP_FORMATION
FR_RETURN_RESULT	The tool must return the result as an attachment to users.	UR_GAIN_RESULT
FR_DATA_STORE	No personal information is ever stored on the tool.	UR_SAFETY
FR_DATA_REMOVE	All data is removed after the session is finished.	UR_SAFETY

Figure 4. Functional Requirements

<u>Non-functional Requirements</u>

Requirement ID	Requirement detail	User Requirement
NFR_MANUAL	Simple and clearly instructions will be listed on the tool.	UR_DIFFICULTY
NFR_LOADING	The tool should give progress feedback to the user, while the groups are formed.	UR_DIFFICULTY

Figure 5. Non-Functional Requirements

4 Design and Implementation

4.1 System Architecture

The web-based tool will open an instruction which explains how to use the tool (NFR_MANUAL). The user is asked to select an algorithm between a Genetic algorithm and a clustering algorithm which they want to use to form the groups (FR_CHOOSE_ALGORITHM). The user is shown an input box that requires the number of groups they want to form (FR_INPUT_GROUP_NUMBER). As required by FR_INPUT_TYPE, the input field detailed below:

- Group number must be integer.
- Group number needs to be at least 2.
  (Form at least 2 groups)
- Group number must be divisible by total number of students. (Every group will have the same number of students)

The user is required to upload a csv file containing students' details (FR_UPLOAD_DATA), required by FR_CSV_DATA:

- Student ID must be unique.
- There should be at least 4 students .
- There must not be a column where all students' characteristics have the same value.

Pressing “Choose file” button will take the user to their local desktop. Once they make the selection of the csv file, pressing “Submit” button allows the backend server to check if all inputs are filled and in the correct formats (FR_INPUT_REQUIRED). If not, an error message will be displayed asking for retype. If all information is corrected uploaded to the tool, the data will be processed and the selected algorithm will create the groups (FR_IMPLEMENTATION). While wating for the results, a loading icon will be displayed so that users know what is going on (NFR_LOADING). After groups are formed, a new csv file will be downloaded to user's local desktop automatically (FR_RETURN_RESULT). This is the end of group formation. No personal information is ever stored on the tool (FR_DATA_STORE) and all data will be removed once the session ends (FR_DATA_REMOVED) as required by UR_SAFETY. Figure 6 is the activity diagram of the process of group formation of the tool:

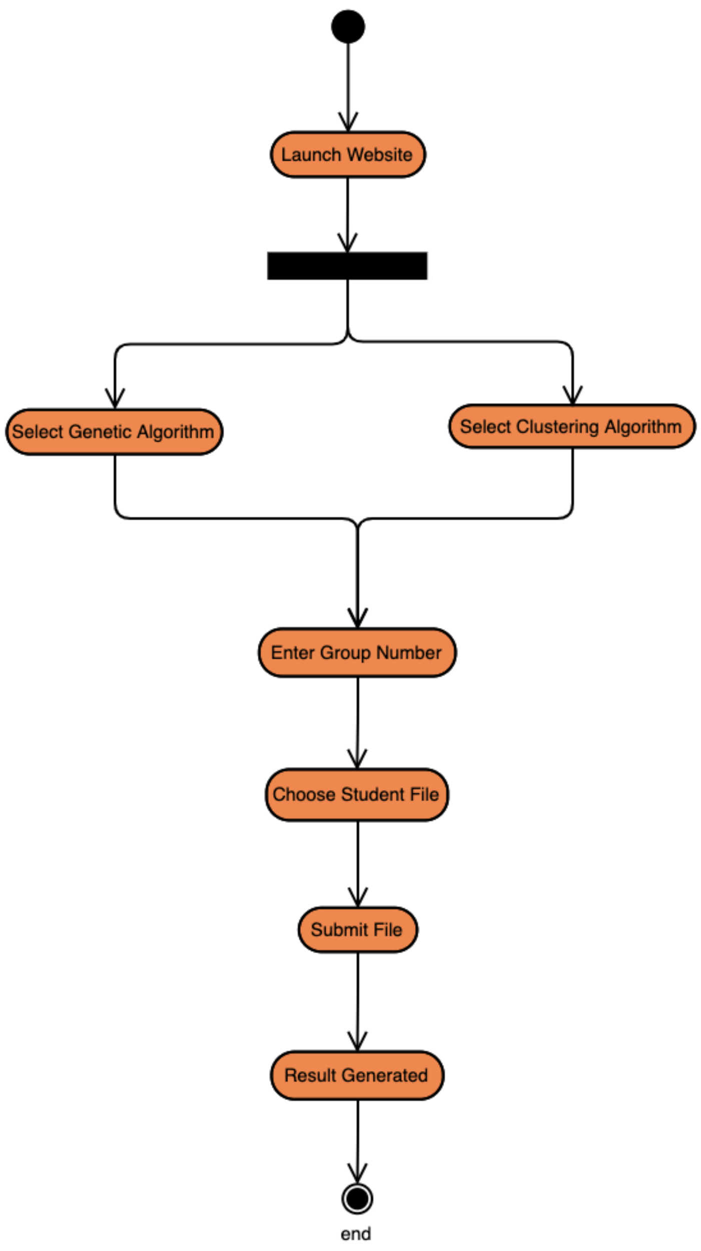


Figure 6. Activity diagram of the process of group formation

The system is a web-based tool using flask web framework which provides reusable code and extensions for common operations so it allows developers easily maintenance, testing and debugging [26]. The users are allowed to interact with the application using browsers on their devices. With internet connection, the web-based tool can be accessed easily anywhere. Our project aims to provide a web-based tool which can automate the student group formation process.

The system was built step by step. In the first version, we only focused on the implementation of the main algorithms - Genetic algorithm and clustering-based algorithm for assigning students to collaborative groups. The backend for the implementation of algorithms is developed with Python. The application's frontend is based on HTML, JavaScript, CSS and React. Finally, the application was deployed in the could application platform Heroku: http://groupformation.herokuapp.com/

The general system architecture has three layers:

- presentation tier: The end users interact with the application. It is used to display information and collect data from the users.
- Application tier: Implementation of business logic and process the data collected from the presentation tier.
- Data tier: Access data components.

Figure 7 is the Flow chart of the group formation system:

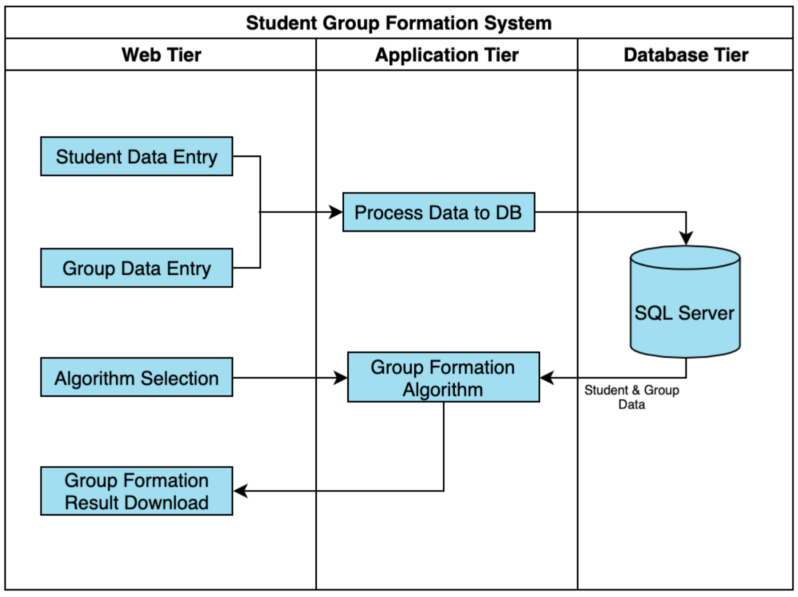


Figure 7. Flow chart of the system

The system was built with simple interface to allow data entry and the generation of group formation result. The system contains the following components:

- Web Interface.
- Process data to DB and process group formation algorithm at backend classes.

- Database with student information and group information.
- Web-based interface will download the result as an attachment automatically once groups are formed.

4.2 Web Interface

The web interface [see Figure 16 in Appendix A] uses HTML, JavaScript, CSS and React. The system can be accessible on Chrome, Firefox and IE browser. The following contents are the website navigation:

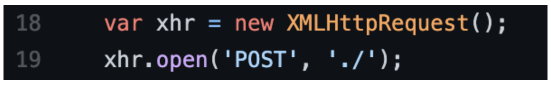
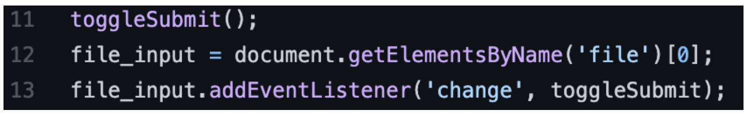
1) An introduction about the tool.
2) Algorithm selection to choose which algorithm to be implemented.
3) Enter the number of groups that need to be formed.
4) Upload csv file that contains students' information.
5) A loading icon.

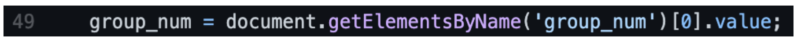
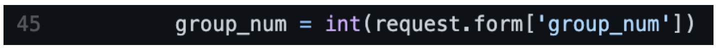
The web-based tool only has one page. The web interface will require group number input and student data input having students' IDs and their characteristics. Students' data needs to be formatted as a csv file. The user will also need to select which algorithm is going to be implemented to form the groups. Two options are provided on the page: genetic algorithm and clustering algorithm. The system then calculates the data and runs the algorithm at the backend. While waiting for the result, a loading icon will be displayed on the web page. The system will generate the group formation result as a csv on the web page which will be downloaded automatically. Find the instructions of using the tool [Figure 15] and an example of the use of the tool [Figure 17] [Figure 18] in Appendix A.

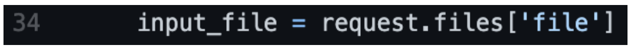
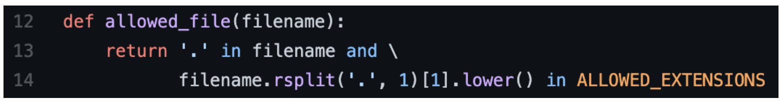
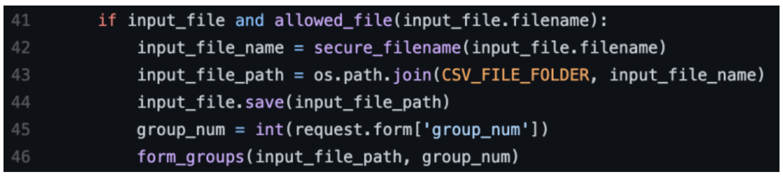
4.3 Feature Implementation

Find Figure 8 the five aspects of the tool that were identified as the main features, and how they were implemented. This is not an exhaustive specification of the entire solution however this does show the process used to develop the front/ back end functionality of the solution to accomplish its given tasks.

Feature	How it was implemented

Algorithm selection	We used *XMLHttpRequest* to transfer data between the web browser and web server by utilising a HTML *POST* request method.
	
	`var xhr = new XMLHttpRequest();`
	`xhr.open('POST', './');`
	Used JavaScript *getElementsByName()* and *change* event to select which algorithm is going to be implemented, attached an event handler to it by using *addEventListener method.*
	
	`toggleSubmit();`
	`file_input = document.getElementsByName('file')[0];`
	`file_input.addEventListener('change', toggleSubmit);`

Input group number	We created a view which accepts a *POST* request, using Flask the global *request* object to access request data.
	
	`group_num = document.getElementsByName('group_num')[0].value;`
	
	`group_num = int(request.form['group_num'])`

Upload file	We defined an HTML form with a file field in it, and access to submitted form filed in the *request.files* dictionaries which is provided by Flask.
	
	`input_file = request.files['file']`
	Using *FileAllowed* to allow extensions of the file.
	
	`def allowed_file(filename):`
	`    return '.' in filename and \`
	`           filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS`
	The file will be saved, and the algorithm will access the path of the file.
	
	`if input_file and allowed_file(input_file.filename):`
	`    input_file_name = secure_filename(input_file.filename)`
	`    input_file_path = os.path.join(CSV_FILE_FOLDER, input_file_name)`
	`    input_file.save(input_file_path)`
	`    group_num = int(request.form['group_num'])`
	`    form_groups(input_file_path, group_num)`

Result downloa d	Once the groups are formed, the result will be sent back to the user and output as an attachment

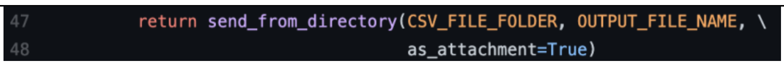
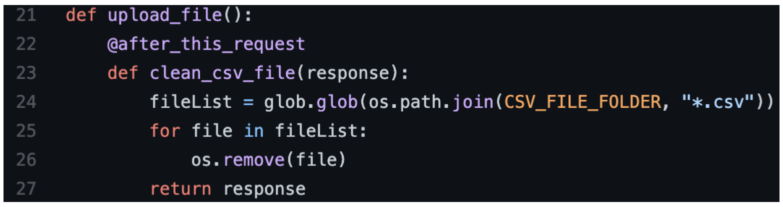
	 `return send_from_directory(CSV_FILE_FOLDER, OUTPUT_FILE_NAME, \` `as_attachment=True)`
Delete file	The uploaded csv file will be deleted once the file is uploaded to ensure data security.  `def upload_file():` `    @after_this_request` `    def clean_csv_file(response):` `        fileList = glob.glob(os.path.join(CSV_FILE_FOLDER, "*.csv"))` `        for file in fileList:` `            os.remove(file)` `        return response`

Figure 8. Features implementation

4.4 Data processing

We determined three main factors that will affect the level of collaboration: learning style, cognitive competence and basic knowledge level according to Literature on grouping criteria [see 2.1.2].

1) Soloman's Index of Learning Styles online survey instrument is often used to measure learning style on 4 dimensions (active/reflective, sequential/global, sensing/intuitive and visual/verbal) [23]. Questionnaires are available online [24]. Students will answer 44 a-b questions and submit the survey, their results will be reported back immediately on the website to be copied or downloaded, and they will not be stored.
2) Cognitive competence test is to measure a student's capability of learning a new process or fulfilling a complex task. Test often includes numerical, abstracting reasoning and verbal questions. Various online tools can help the instructor to identify the students with the right potential and abilities to succeed in the tasks.
3) Basic knowledge level measurement is based on students' past exams or assessments results. They can be gained from department database. We take the average marks of the relevant modules for each student. These data are not fixed, they vary term to term. The instructor may change the newest results in order to match students' levels correctly.

4.5 Algorithms

In this session, we discuss a clustering-based algorithm and a genetic algorithm to solve the group formation problem. However, the algorithms are interchangeable. More algorithms can be added on the tool by simply implementing the algorithm selection feature.

A. Clustering-Based Algorithm

A clustering-based algorithm is proposed for the group formation problem in this report. Firstly, the characteristics of students will be quantified. Then the algorithm will assign students that have the most similarity into the same group, in total there will be *k* groups and each of the groups represents a cluster. The algorithm will assign *k* students from *k* clusters randomly to each of the *k* collaborative groups. After each iteration, *k* student that has been allocated will be removed and the centroid will be then updated. This process repeats until all students are assigned. In this case, students with similar abilities will be separated into different groups. An example of clustering algorithm to group six students is given in Appendix Figure 12-14.

The steps of the procedure are:
(1) Set the variable *stu* as the initial total number of students.
(2) Calculate the mean of each characteristic of all *stu* students and generate an array containing these means. Use this array as the centroid of the cluster.
(3) Find the closet *k* students to the centroid by computing their Euclidean distance between *stu* students and the centroid.
(4) Assign these *k* students to different *k* collaborative groups randomly.
(5) *stu = stu – k,* i.e., remove the above *k* students from *stu.*
(6) Repeat step 2 to step 5, until *stu = 3*.
(7) Assign the last k students to k groups.

Pseudocode of the algorithm:

```
Input:
Sm = {c1, c2, c3,... cn}      // Student m with n characteristics
K      // Number of desired groups
Output:
K     // set of groups
Algorithm:
1: stu = m
2: repeat
3:     for c = 1 to n do
4:        calculate the total mean of m students
5:              for s = 1 to m do
```

```
6:                  calculate the Euclidean Distance to the total mean
7:             assign the closet k students to k groups randomly
8:     m = m – k
9: until stu = k
10: assign the final k students to k groups
```

B. Genetic Algorithm

1) Student Representation

The genetic algorithm is an implementation based on the approach presented by Moreno et al [3]. Each student is represented as an array S_m = {c1, c2, c3}. The characteristics will be normalized to 0-1 range. A feasible solution is said to be an individual, which is represented by student IDs.

2) Individual Representation

An individual contains more than one array, that is at least two groups will be formed. Students are in the same group if their IDs belong to the same array. For instance, if there are 6 students, and 2 groups are needed, each group will have 3 different students, an individual is represented by:

[[*1, 3, 5*],
  [*4, 2, 6*]]

3) Fitness function

A fitness function is defined to measure the quality of each individual. Since our goal is to obtain intra-heterogenous and inter-homogeneous groups, we will measure the homogeneity for each individual. The main steps for the fitness calculation are:

1) Calculate the total mean for each of the characteristics of all students: TM = { $\overline{c1}$, $\overline{c2}$, $\overline{c3}$}
2) For each group in an individual, the mean of each attribute must be calculated. The individual mean is calculated:
   IM= { $\overline{c1}$, $\overline{c2}$, $\overline{c3}$ (of S_1, S_3, S_5),
   $\overline{c1}$, $\overline{c2}$, $\overline{c3}$ (of S_4, S_2, S_6)}
3) The sum of squared difference between each characteristic's individual mean and characteristic's total mean is calculated:
   Fitness = ($\overline{c1}$™ - $\overline{c1}^{IM}$)^2 + ($\overline{c2}$™ - $\overline{c2}^{IM}$)^2 + ($\overline{c3}$™ - $\overline{c3}^{IM}$)^2 + …

The smaller the fitness score, the more similar each group of such an individual will be. In order to be convenient, we take the reciprocal of this score. In this case, the higher fitness score individual will be saved.

4) Genetic Operators

At the start of the algorithm, we create a population of N individuals randomly. Then we calculated the fitness score for each individual in that population. According to the fitness score, we apply the probability to all individuals by using roulette wheel, the higher the fitness score, the higher probability that individual will be selected. This is called selection and it will repeat until a percentage *x* of the population is selected. Later the crossover operator will be performed in order to produce “children”, namely, better individuals. The operator chooses multiple random points between the chromosomes of two parents. The genes before the random points will remain, the genes after the random points will be removed and the empty space will be filled in the order the genes appear in the second parent.

For example, we have parent1 and parent2, the random points are {2, 1}:

Parent1: [[1, 2, 3], parent2: [[4, 5, 1],
[4, 5, 6]] [3, 2, 6]]

Child1: [[1, 2, _],
[4, _, _]]

Student 3, 5, 6 are missing after removing genes that located after random points, then we look at parent2, the removed genes appear in the order of 5, 3, 6 from top to the bottom, therefore we fill child1 with this order:

Child1: [[1, 2, 5],
[4, 3, 6]]

The next step is that the remaining *1-x* percentage of each new generation will be obtained by crossover operator and parents will be selected. Then the mutation operator will be performed on these parents. The process is repeated generation to generation until it reaches a terminated condition, here we define a maximum number of iterations.

5 Results and Evaluation

Research Question 1: Does the proposed genetic algorithm and the proposed clustering algorithm generate groups with smaller variance than a random search algorithm?

Research Question 2: Does the proposed genetic algorithm and the proposed clustering algorithm generate groups with smaller execution time than a random search algorithm?

5.1 Experiment method

As prior literature review determined the important criteria [see 4.4], we consider 3 factors to assign students to groups: Learning style, cognitive competence and basic knowledge level, where learning style measures 4 dimensions: Active/reflective, sequential/global, sensing/intuitive and visual/verbal, and each of them corresponds to a characteristic. Therefore, each student could be represented by an array: S_m = {c1, c2, c3, c4, c5, c6}, where c1 is active/reflective characteristic, c2 is sequential/global characteristic, c3 is sensing/intuitive characteristic, c4 is visual/verbal characteristic, c5 is cognitive competence characteristic, c6 is basic knowledge level characteristic.

In order to evaluate the performance of the algorithms, i.e., the extent of heterogeneity of the result, we will conduct several experiments by using self-generated student data. We will use Excel RANDBETWEEN() function to generate numbers which represent student characteristics.

After forming the groups, the standard deviation for each characteristic of students who are in the same group will be calculated, then we sum up the standard deviations of all characteristic in that group, for better understanding, we call it group standard deviation sum. Finally we take the average of all groups standard deviation sums to represent the variance of groups in that cohort. Higher standard deviation indicates larger individual differences and higher level of heterogeneity among students, which is good as student can learn multiple aspects from their peers [see 2.1.1]. The homogeneity of the whole cohort will also be measured by calculating the normal distribution of group standard deviation sums, we want this to be as small as possible as we want fairness for all groups.

The scalability of the algorithms will be evaluated in terms of execution time of the algorithms by conducting experiments with different number of students. The fitness scores will be calculated by the fitness function which was discussed in 4.5)B)3) after the groups are formed.

5.2 Experiment setup

We generate 6 datasets to test the performance of algorithms. Student size should be realistic hence we generated 6 cohort of 100, 200, 300, 400, 500 and 600 students respectively. As prior literature review determined that the optimal group size should be around 4 [see 2.1.3], hence the datasets will be divided into 25 groups of 4, 50 groups of 4, 75 groups of 4, 100 groups of 4, 125 groups of 4 and 150 groups of 4 respectively. The student data will be generated by using Excel RANDBETWEEN(), the range of c1, c2, c3 and c4 is set to be 0-10, c5 ranges from 0 to 1 and c6 ranges from 0 to 100. Our goal is to produce heterogeneous groups.

To test the scalability of the algorithms, we will run the algorithms on the above datasets, the algorithm stops when it outputs a solution. The execution time and fitness score of the algorithms will be recorded. Our goal is to achieve short execution time and high fitness score.

Moreno et al [3] suggested using low mutation rate. After several runs of the genetic algorithm, the optimal parameters are defined in Figure 9, as they got a faster convergence:

Population size	100
Individual mutation rate	0.1
Gene mutation rate	0.05
Crossover rate	0.6
Maximum generation number	1000

Figure 9. The parameters of genetic algorithm

5.3 Performance of Results

Session 5.3.1 refers to research Question 1, session 5.3.2 refers to research Question 2.

5.3.1 Performance

			GA		**Clustering**		**Random**	
Data-set	Student number	Group number	Average group SD	cohort SD	Average group SD	Cohort SD	Average group SD	Cohort SD
A	100	25	41.78	11.20	41.34	9.35	38.77	10.37
B	200	50	40.93	10.89	41.05	11.35	39.06	10.37
C	300	75	40.92	10.34	40.25	8.99	39.72	9.72
D	400	100	42.56	10.46	41.82	10.09	38.75	11.35
E	500	125	40.81	11.09	41.02	11.38	39.94	10.50
F	600	150	43.76	10.76	43.63	11.08	39.58	9.64

Figure 10. Experiment results

Dataset A-F represent 6 cohorts, they were divided into 25, 50, 75, 100, 125 and 150 groups respectively. The results are shown in Figure 10, the cohort normal distribution of dataset C is shown in Appendix Figure 19 - 21.

We can see that both GA and Clustering produced results with an average group standard deviation above 40, whereas the random method produced a result with a standard deviation ranging from 38.75 to 39.94.

It could be seen that both Genetic algorithm and clustering algorithm generated high average standard deviations, indicating that students in their groups have high diversity, which achieved our goal of obtaining heterogeneity inside groups. The variances generated by the proposed methods are bigger than the results generated by a random search algorithm.

In terms of cohort normal distribution, the GA produced a cohort variance from 10.34 to 11.20, clustering algorithm produced 8.99 to 11.38, and random generation produced variance ranging from 9.64 to 11.35. It could be observed that the proposed clustering algorithm and random generation produced better inter-homogeneous groups than GA.

Therefore, it could be observed the proposed genetic algorithm and the proposed clustering algorithm generate groups with bigger group variance than a random search algorithm. However, the proposed algorithms do not generate better inter-homogeneous groups, i.e., smaller cohort variance, than a random search algorithm.

5.3.2 Scalability

Students	**GA**		**Clustering**		**Random**	
	Time	Fitness	Time	Fitness	Time	Fitness
100	20.691	23.09	0.051	23.17	0.031	5.03
200	113.460	24.59	0.166	24.48	0.162	7.62
300	150.547	29.63	0.309	26.39	0.217	4.92
400	203.726	27.40	0.412	27.50	0.374	23.81
500	260.832	24.82	0.529	21.83	0.483	25.96
600	280.573	21.34	0.580	20.62	0.510	17.48

Figure 11. Experiment results of scalability

The proposed three algorithms were implemented in the Python programming language using Visual Studio Code environment and all runs were done on MacBook with 3.1 GHz Dual-Core Inter Core i5.

Results are shown in Figure 11. The GA computation time became larger when the size of students grows, whereas the clustering algorithm computed good fitness result at a low computation time. The random generation method produced unstable fitness score in the experiments at a low computation time.

Therefore, the proposed genetic algorithm and the proposed clustering algorithm do **NOT** generate groups with smaller execution time than a random search algorithm.

6 Conclusion

The report proposes an effective approach to assign students to groups. The optimal group size is proposed to be 4, and heterogenous groups are considered to have more benefit for group assessments as group members will have different knowledge level and different perspectives of learning styles to increase the overall learning experience. The arbitrary number of student characteristics provide a flexibility for the educators so that they can adjust the grouping criteria according to the types of group assessments.

Regarding the goal of implementing a genetic algorithm and a clustering algorithm to formulate heterogeneous groups for students, the system is very effective. To evaluate the performance and scalability of the proposed methods, several experiments were conducted to compare the standard deviation of the groups that were generated by genetic algorithm, clustering algorithm and random generation method. The results show that the average standard deviations of the genetic algorithm and the clustering algorithm are higher than the random-generation approach, which indicated higher individual difference in groups that were generated by our proposed methods. And while the size of students gets larger, genetic algorithm has higher computation cost than the clustering algorithm and random generation method. In conclusion, both the genetic algorithm and the clustering algorithm can produce a good result within limited time.

The automated group formation online tool will reduce the time and effort needed to create student teams for an educator, where the educator only needs to define the grouping criteria and the group size. It estimates that issuing the surveys to students, collecting data from students and creating groups will take 30 minutes. It saves time for the educator especially when the size of students is large.

Due to Covid, it is hard to collect real students' data, therefore this project used self-generated data to evaluate the algorithms. There are only six cohorts being tested for the performance of the algorithms, more experiments could be conducted to calculate and verify the variance of groups. In future work, we also expect to monitor students' marks after grouping with our tool, as our experiments used self-generated data, which may not represent a real case. Moreover, it would be useful to create homogeneous groups to give the user more flexibility when choosing the type of groups. More algorithms can be added to solve the group formation problem to make our system better.

Appendix A

Example of clustering algorithm

Assume there are *6* students with *3* characteristics who will be divided into 2 groups. Each student is represented by an array S_m = {c1, c2, c3}, where each characteristic c is a numerical value [Figure 12]:

	C1	C2	C3
Student 1	1	0.2	75
Student 2	5	0.8	73
Student 3	8	0.5	82
Student 4	6	0.3	64
Student 5	2	0.6	62
Student 6	3	0.7	78

Figure 12. Example of clustering algorithm

Normalize data to fit 0-1 range [Figure 13]:

	C1	C2	C3
Student 1	0	0	0.65
Student 2	0.571	1	0.55
Student 3	1	0.429	1
Student 4	0.714	0.143	0.1
Student 5	0.143	0.571	0
Student 6	0.286	0.714	0.8
Total Mean	0.452	0.476	0.517

Figure 13. Normalization of data

Calculate Euclidean distance to the total mean {0.452, 0.476, 0.517}:
S1 = {0, 0, 0.65}, ed1 = 0.6698
S2 = {0.571, 1, 0.55}, ed2 = 0.5384
S3 = {1, 0.429, 1}, ed3 = 0.732
S4 = {0.714, 0.143, 0.1}, ed4 = 0.5945
S5 = {0.143, 0.571, 0}, ed5 = 0.6098
S6 = {0.286, 0.714, 0.8}, ed6 = 0.4053
The closet 2 students to the total mean are S2 and S6, assign them to 2 collaborative groups randomly: K1 = {S2}, K2 = {S6}.

Now we delete S2 and S6 from the lists and recalculate the total mean and Euclidean distance [Figure 14]:

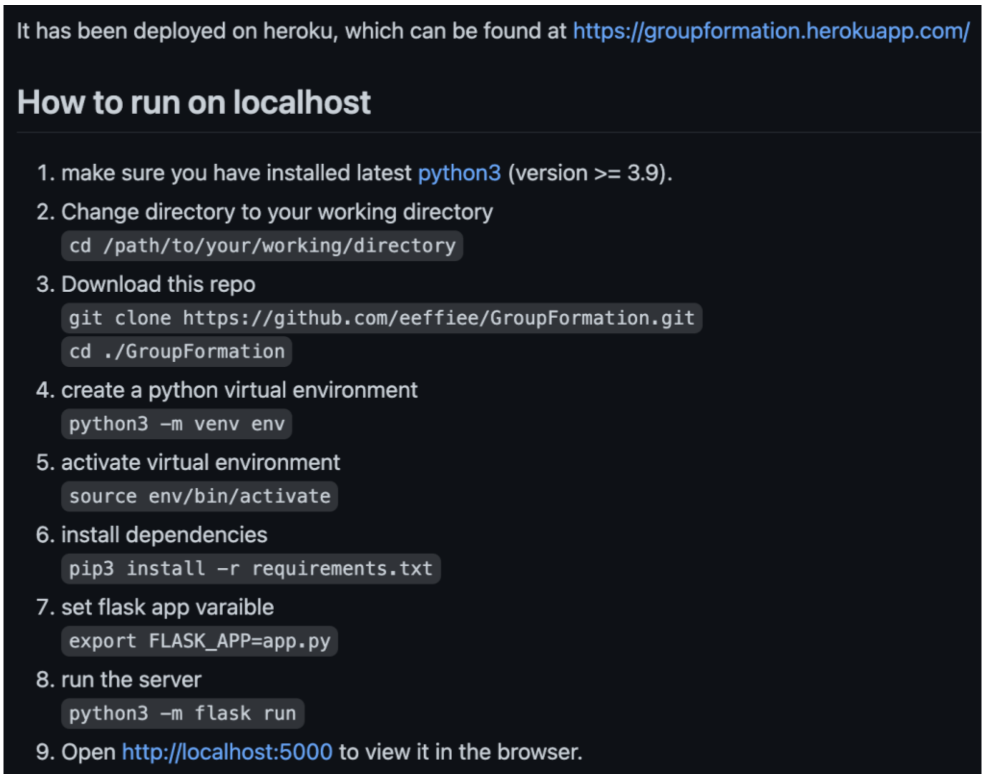
	C1	C2	C3
Student 1	0	0	0.65
Student 3	1	0.429	1
Student 4	0.714	0.143	0.1
Student 5	0.143	0.571	0
Total Mean	0.464	0.286	0.438

Figure 14. Example of clustering algorithm

Calculate Euclidean distance to the total mean {0.464, 0.286, 0.438}:
S1 = {0, 0, 0.65}, ed1 = 0.5848
S3 = {1, 0.429, 1}, ed3 = 0.7897
S4 = {0.714, 0.143, 0.1}, ed4 = 0.4441
S5 = {0.143, 0.571, 0}, ed5 = 0.6133

The closet 2 students to the total mean are S1 and S4, assign them to 2 collaborative groups randomly: K1 = {S2, S4}, K2 = {S6, S1}.
Now we delete S1 and S4 from the lists, we have S3 and S5 left. Therefore we assigned them randomly : K1 = {S2, S4, S3}, K2 = {S6, S1, S5}. All students are assigned and the final result is:
Group1: Student 2, Student 3, Student 4
Group2: Student 1, Student 5, Student 6

Instruction of using the tool



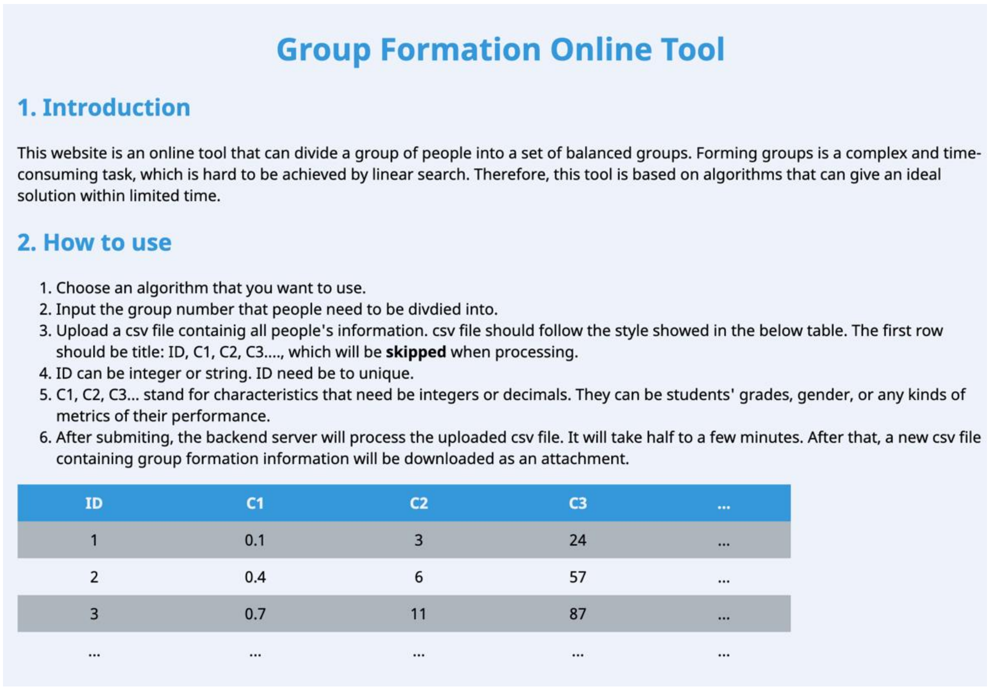
It has been deployed on heroku, which can be found at https://groupformation.herokuapp.com/

How to run on localhost

1. make sure you have installed latest python3 (version >= 3.9).
2. Change directory to your working directory
   ```
   cd /path/to/your/working/directory
   ```
3. Download this repo
   ```
   git clone https://github.com/eeffiee/GroupFormation.git
   cd ./GroupFormation
   ```
4. create a python virtual environment
   ```
   python3 -m venv env
   ```
5. activate virtual environment
   ```
   source env/bin/activate
   ```
6. install dependencies
   ```
   pip3 install -r requirements.txt
   ```
7. set flask app varaible
   ```
   export FLASK_APP=app.py
   ```
8. run the server
   ```
   python3 -m flask run
   ```
9. Open http://localhost:5000 to view it in the browser.

Figure 15. Instructions of using the tool

Website interface



Group Formation Online Tool

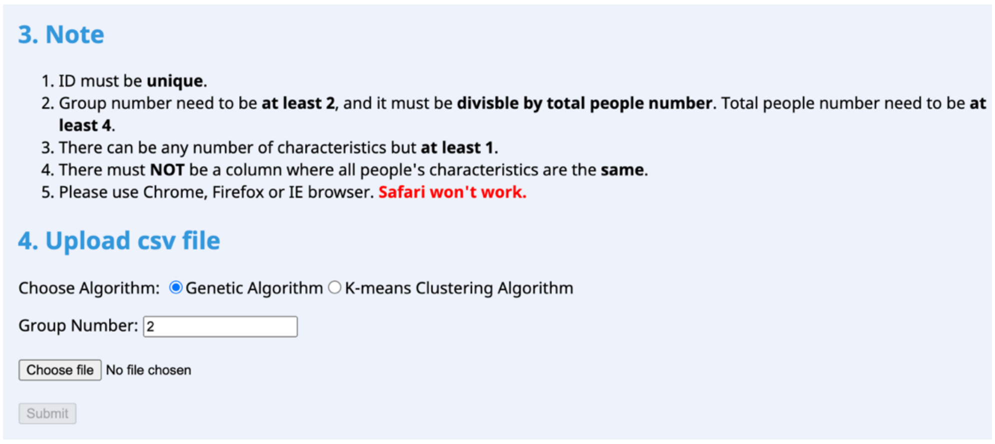
1. Introduction

This website is an online tool that can divide a group of people into a set of balanced groups. Forming groups is a complex and time-consuming task, which is hard to be achieved by linear search. Therefore, this tool is based on algorithms that can give an ideal solution within limited time.

2. How to use

1. Choose an algorithm that you want to use.
2. Input the group number that people need to be divdied into.
3. Upload a csv file containig all people's information. csv file should follow the style showed in the below table. The first row should be title: ID, C1, C2, C3...., which will be **skipped** when processing.
4. ID can be integer or string. ID need be to unique.
5. C1, C2, C3... stand for characteristics that need be integers or decimals. They can be students' grades, gender, or any kinds of metrics of their performance.
6. After submiting, the backend server will process the uploaded csv file. It will take half to a few minutes. After that, a new csv file containing group formation information will be downloaded as an attachment.

ID	C1	C2	C3	...
1	0.1	3	24	...
2	0.4	6	57	...
3	0.7	11	87	...
...	...	...	...	...



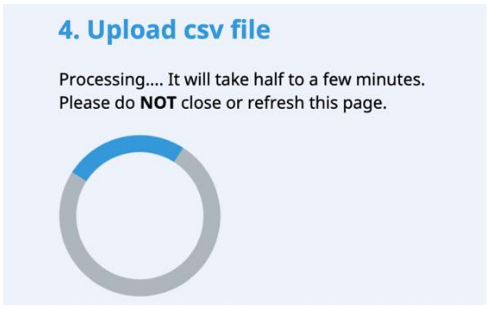
3. Note

1. ID must be **unique**.
2. Group number need to be **at least 2**, and it must be **divisble by total people number**. Total people number need to be **at least 4**.
3. There can be any number of characteristics but **at least 1**.
4. There must **NOT** be a column where all people's characteristics are the **same**.
5. Please use Chrome, Firefox or IE browser. **Safari won't work.**

4. Upload csv file

Choose Algorithm: [✓ Genetic Algorithm] K-means Clustering Algorithm

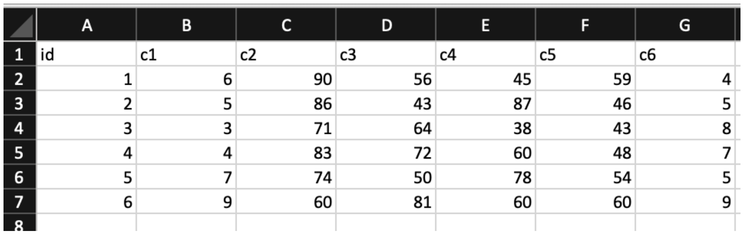
Group Number: 2



4. Upload csv file

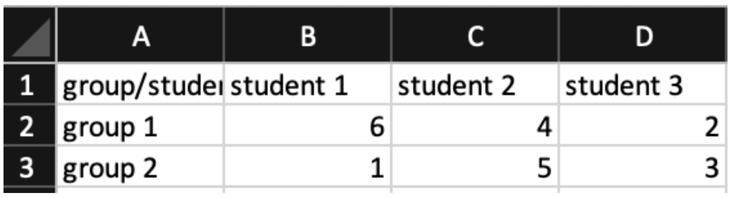
Processing.... It will take half to a few minutes.
Please do **NOT** close or refresh this page.

Figure 16. The website of group formation tool



	A	B	C	D	E	F	G
1	id	c1	c2	c3	c4	c5	c6
2	1	6	90	56	45	59	4
3	2	5	86	43	87	46	5
4	3	3	71	64	38	43	8
5	4	4	83	72	60	48	7
6	5	7	74	50	78	54	5
7	6	9	60	81	60	60	9
8							

Figure 17. Example csv file



	A	B	C	D
1	group/studei	student 1	student 2	student 3
2	group 1	6	4	2
3	group 2	1	5	3

Figure 18. Result using genetic algorithm

Cohort normal distribution

Where x-ais refers to the sum of standard deviation of all characteristics for each group, y-ais refers to its normal distribution for each group in a cohort.

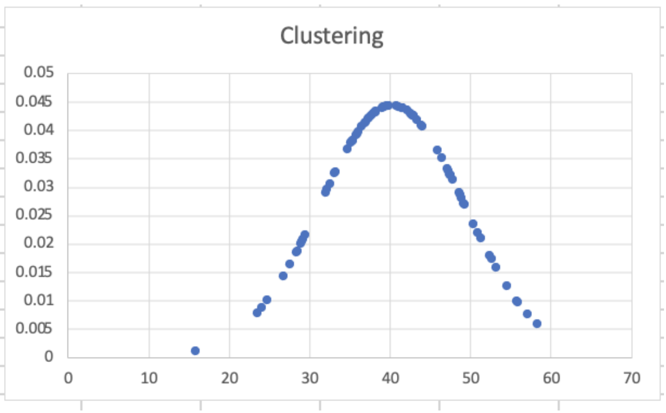


Figure 19. Cohort normal distribution

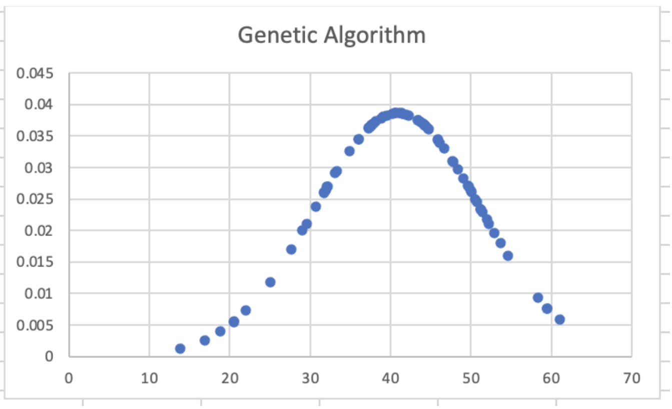


Figure 20. Cohort normal distribution

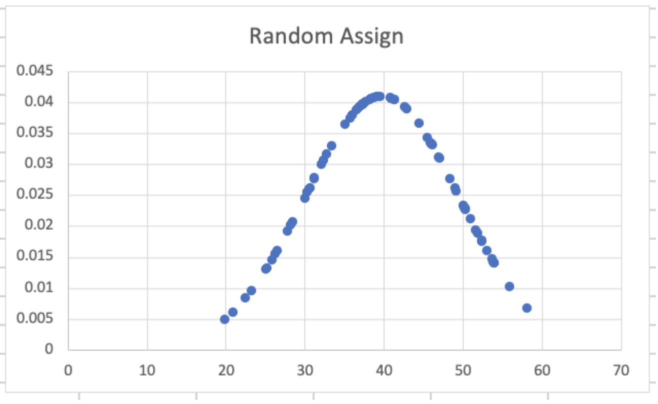


Figure 21. Cohort normal distribution

Bibliography

[1] J. L. Smith and R. M. Spindle, “The impact of group formation in a cooperative learning environment,” *Journal of Accounting Education*, vol. 25, no. 4, pp. 153-167, 2007. [Online]. Available: https://doi.org/10.1016/j.jaccedu.2007.09.002. [Accessed: 1 June 2021]

[2] Z. C. Ani, et al., “A Method for Group Formation Using Genetic Algorithm,” *International Journal on Computer Science and Engineering*, vol. 02, pp. 3060-3064, January 2010. [Online]. Available: https://www.researchgate.net/publication/229035954_A_Method_for_Group_Formation_Using_Genetic_Algorithm. [Accessed: 1 June 2021]

[3] J. Moreno, et al., “A genetic algorithm approach for group formation in collaborative learning considering multiple student characteristics,” *Computers & Education*, vol. 58, no. 1, pp. 560-569, January 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360131511002284. [Accessed: 1 June 2021]

[4] S. Graf and R. Bekele, “Forming Heterogeneous Groups for Intelligent Collaborative Learning Systems with Ant Colony Optimization,” *Intelligent Tutoring Systems*, vol. 4053, pp. 217-226, 2006. [Online]. Available: https://doi.org/10.1007/11774303_22. [Accessed: 1 June 2021]

[5] J. Wang and K. Kojima, “Exploring heterogeneous grouping strategies from the learning analytic perspective,” *International Journal of Learning Technologies and Learning Environments*, vol. 2, no. 2, pp. 21-34, November 2019. [Online]. Available: http://www.iaiai.org/journals/index.php/IJLTLE/article/view/479. [Accessed:1 June 2021]

[6] Y. Pang, et al., "A Clustering-Based Grouping Model for Enhancing Collaborative Learning," *2014 13th International Conference on Machine Learning and Applications*, pp. 562-567, December 2014. [Online]. Available: https://ieeexplore.ieee.org/document/7033177 [Accessed: 1 June 2021]

[7] Z. Ge, et al., "A Literature Review of Grouping Solutions in Collaborative Learning," *2018 IEEE International Conference on Progress in Informatics and Computing (PIC)*, pp. 393-397, 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8706326. [Accessed: 1 June 2021]

[8] L. Razmerita and A. Brun, “Collaborative Learning in Heterogeneous Classes: Towards a Group Formation Methodology,” *International Conference on Computer Supported Education*, May 2011. [Online]. Available: https://www.researchgate.net/publication/50853021_Collaborative_Learning_in_Heterogeneous_Classes_Towards_a_Group_Formation_Methodology. [Accessed: 1 June 2021]

[9] C. E. Christodoulopoulos and K. A. Papanikolaou, “A Group Formation Tool in an E-Learning Context,” *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, pp. 117-123, October 2007. [Online]. Available: https://ieeexplore.ieee.org/document/4410368. [Accessed: 1 June 2021]

[10] P. K. Imbrie, et al., “Genetic Algorithm Optimization of Teams for Heterogeneity,” *2020 IEEE Frontiers in Education Conference (FIE)*, pp. 1-5, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9274243. [Accessed: 1 June 2021]

[11] J. Patel and M. M. Mujahedi (2017, May. 5), *Automated Team Assignment System using Greedy Algorithm for Efficient Student Group Formation*, CSIS [Online]. Available: http://csis.pace.edu/~ctappert/srd2017/2017PDF/d12.pdf [Accessed: 1 June 2021]

[12] M. H. Dlab and N. H. Bozic, “Implementation of Group Formation Algorithms in the ELARS Recommender System,” *International Journal of Emerging Technologies in Learning (iJET)*, vol. 12, no. 11, November 2017. [Online]. Available: https://www.researchgate.net/publication/321109006_Implementation_of_Group_Formation_Algorithms_in_the_ELARS_Recommender_System. [Accessed: 1 June 2021]

[13] A. Gogoulou, et al. (2007), *Forming Homogeneous, Heterogeneous and Mixed Groups of Learners*, Citeseerx [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.136.485&rep=rep1&type=pdf [Accessed: 1 June 2021]

[14] C. E. Christodoulopoulos and K. A. Papanikolaou (2007, Januaary), *Investigation of Group Formation using Low Complexity Algorithms*, Citeseerx [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.174.5284&rep=rep1&type=pdf [Accessed: 1 June 2021]

[15] D. Lambić, et al., “A novel metaheuristic approach for collaborative learning group formation,” *Journal of Computer Assisted Learning*, vol. 34, pp. 907-916, August 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1111/jcal.12299. [Accessed: 1 June 2021]

[16] S. Luke, *Essentials of Metaheuristics*, 2016. [Online]. Available: https://cs.gmu.edu/~sean/book/metaheuristics/Essentials.pdf [Accessed: 1 June 2021]

[17] K. Hussain, et al., “Metaheuristic research: a comprehensive survey,” *Artificial Intelligence Review*, vol. 52, pp. 2191-2233, December 2019. [Online]. Available: https://link.springer.com/article/10.1007/s10462-017-9605-z. [Accessed: 1 June 2021]

[18] B. Kizielewicz and W. Sałabun, “A New Approach to Identifying a Multi-Criteria Decision Model Based on Stochastic Optimization Techniques,” *Symmetry*, vol. 12, no. 9, pp. 1551, September 2020. [Online]. Available: https://www.mdpi.com/2073-8994/12/9/1551. [Accessed: 1 June 2021]

[19] B. Chopard and M. Tomassini, “Performance and Limitations of Metaheuristics,” *Natural Computing Series*, pp. 191-203, January 2018. [Online]. Available: https://www.researchgate.net/publication/328690647_Performance_and_Limitations_of_Metaheuristics. [Accessed: 1 June 2021]

[20] Y. Lin, et al., “An automatic group composition system for composing collaborative learning groups using enhanced particle swarm optimization,” *Computer & Education*, vol. 55, no. 4, pp. 1483-

1493, December 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360131510001740. [Accessed: 1 June 2021]

[21] H. Sadeghi and A. A. Kardan, "A novel justice-based linear model for optimal learner group formation in computer-supported collaborative learning environments," *Computers in Human Behavior*, vol. 48, pp. 436-447, July 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0747563215000345. [Accessed: 1 June 2021]

[22] R. Cavanaugh, et al., "Automating The Process Of Assigning Students To Cooperative Learning Teams," in *2004 Annual Conference*, Salt Lake City, Utah, June 20- 23, 2004, pp. 9.246.1 - 9.246.14. [Online]. Available: https://peer.asee.org/automating-the-process-of-assigning-students-to-cooperative-learning-teams. [Accessed: 1 June 2021]

[23] R. Brent and R. Felder. "Resources for Teaching and Learning SREM". [Blog entry]. *Rebecca Brent and Richard Felder's Website*. Available: https://educationdesignsinc.com/index-of-learning-styles/ [Accessed: 1 June 2021]

[24] R. M. Felder and B. A. Soloman. *Index of Learning Styles Questionnaire*. NC State University [Online]. Available: https://www.webtools.ncsu.edu/learningstyles/ [Accessed: 1 June 2021]

[25] A. Chilakapati. "A Flask Full of Whiskey (WSGI)". 16 February 2020. [Blog entry]. *Data Exploration*. Available: https://xplordat.com/2020/02/16/a-flask-full-of-whiskey-wsgi/ [Accessed: 1 June 2021]

[26] A. Verma (2020, July. 3). *Python Flask framework and Django framework*. Database Management System [Online]. Available: https://opensource.com/article/18/4/flask [Accessed: 1 June 2021]

[27] N. H. Walker. *An introduction to the Flask Python web framework*. Open Source [Online]. Available: http://www.guidanceshare.com/wiki/Application_Architecture_Guide_-_Chapter_10_-_Presentation_Layer_Guidelines [Accessed: 1 June 2021]

[28] B. Thorgilsson (2018, Feb. 15). *Software Requirements Specification for Super Surveyor*. Pi Technology [Online]. Available: https://nammco.no/wp-content/uploads/2018/05/o04-software-requirements-draft-specification-for-supersurveyor-programme.pdf [Accessed: 1 June 2021]

[29] *Euclidean distance*. Wikipedia [Online]. Available: https://en.wikipedia.org/wiki/Euclidean_distance [Accessed: 1 June 2021]