FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Automatic Group Formation

João Vítor Meireles Chaves

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Ana Paula Rocha

July 24, 2019

Automatic Group Formation

João Vítor Meireles Chaves

Mestrado Integrado em Engenharia Informática e Computação

July 24, 2019

Abstract

Group formation and the way communities group themselves is a task that occurs in the most diverse contexts, such as political movements, all kinds of professional organizations or even work groups of students in classes. This task can become a complex process due to the large number of criteria, more or less quantifiable, that can be used to group people.

This report describes the problem of automatic group formation, which has become a more current problem, due to the importance given to concepts like Collaborative Learning and Learning Management Systems. So, it focuses on the formation of work groups in classes, based on the students' personal data and competences (social and academic), which can be achieved mainly by clustering techniques Therefore, the goal is to use a methodology to create groups that can be heterogeneous or homogeneous, depending on the goal, that is, the type of activity to develop.

There are already some tools that address this topic. They focus on the formation of groups in different environments. The implementations are done by using several methodologies, like clustering (making use of different algorithms) and multi-agent systems. They also differ in terms of the criteria used to group the students.

This work proposes a solution with a similar goal of the existing methodologies. However, it improves the process of clustering by combining the clustering of homogeneous and heterogeneous groups in the same algorithm. The solution implemented starts by briefly explaining the problem of data acquisition and how to overcome it. Then, there is presented the algorithm developed, which is based in one of the most known clustering algorithms, *k-means*.

The project is finalized with some experiments done with the methodology implemented, along with conclusions that demonstrate the efficiency of the algorithm developed.

Resumo

A formação de grupos, e a forma como comunidades se agrupam, é uma tarefa que ocorre nos mais diversos contextos, desde movimentos políticos, organizações profissionais, até grupos de estudantes em aulas. Esta tarefa pode-se tornar complexa, devido ao grande número de critérios, mais ou menos quantificáveis, que podem ser usados para agrupar pessoas.

Este relatório descreve o problema da formação automática de grupos, que se tem tornado cada vez mais um problema corrente, devido à importância dada a conceitos como *Collaborative Learning* e *Learning Management Systems*. Sendo assim, foca-se na formação de grupos de estudantes em unidades curriculares, com base em dados pessoais dos estudantes, bem como competências sociais e académicas. Isto pode ser feito através de técnicas de *clustering*. O objetivo é, portanto, utilizar uma metodologia para criar grupos que podem ser homogéneos ou heterogéneos, dependendo da atividade a realizar.

Já existem algumas ferramentas que tentam resolver este problema, focando-se na formação de grupos em ambientes diferentes. As implementações são feitas utilizando metodologias diferentes, como *clustering* (por meios de algoritmos diferentes) e sistemas multi-agente. As implementações também diferem nos atributos utilizados para agrupar os estudantes.

Este relatório propõe uma solução com finalidade semelhante à das metodologias existentes. No entanto, tenta melhorar o processo de agrupamento de estudantes combinando o *clustering* de grupos homogéneos e homogéneos no mesmo algoritmo. A solução implementada começa por explicar de uma forma breve o problema da aquisição de dados e como ultrapassá-lo. De seguida, é apresentado o algoritmo desenvolvido, o qual é baseado num dos algoritmos de *clustering* mais conhecidos, o *k-means*.

O projeto é finalizado com algumas experiências realizadas com a metodologia implementada, bem como conclusões que demonstram a eficácia do algoritmo desenvolvido.

Acknowledgements

Firstly, I would like to thank my supervisor, Ana Paula Rocha, for all the guidance given throughout the whole project and for the availability to discuss every aspect of the work developed, as well as all the help in the investigation process.

I would also like to thank in a a general way to everyone I met in the past five years at FEUP, everyone whom I learned with and helped me in this experience.

I would like to express my special thanks to my friends, the ones that were always there for me, had to listen to all my complaints and gave me motivation to want to achieve more in this journey.

Last but not the least, I thank my family, in particular my parents and my sister, for all the patience and unconditional support throughout all my years of study.

Thank you all.

João Chaves

"I have not failed.
I've just found 10,000 ways that won't work."

Thomas Edison

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

AGDP	Automated Group Decomposition Program
CL	Collaborative Learning
CSCL	Computer-Supported Collaborative Learning
DOH	Degree of Heterogeneity
EM	Expectation Maximization
FEUP	Faculdade de Engenharia da Universidade do Porto
LMS	Learning Management System
RL	Reinforcement Learning
VARK	Visual, Aural, Read/Write and Kinesthetic

Chapter 1

Introduction

1.1 Context

Group formation and the way communities group themselves to achieve a given goal is a task that occurs in the most diverse contexts. It is an inherent process in the creation of political movements, the formation of professional organizations, even in the formation of students work groups.

However, this process requires that one needs to know what criteria defines a group, in order to define who should be part of it and who shouldn't. It implies that it is also necessary to know the characteristics of each person.

This composes the challenge of defining what criteria defines the best a person and how to quantify and/or qualify some aspects that can not be easily measurable or even known, which can be somehow complex. Another challenge is the one of defining which people should belong to which groups, in order to fulfil the objective required.

1.2 Project

This dissertation addresses the problem of automatically creating work groups of students in classes. In this specific context, there are some things to consider, such as what are the groups intended to achieve. For example, if the goal of an activity is to make the students learn with each other, the best option will be to create groups where the students' skills diverge the most. Therefore, the less skilled ones can learn with the others.

Other important thing to consider is what attributes should qualify the most a student, in order to assign him to a group. In this work, it is intended to understand more about what attributes can define the profile of a student, like academic competences, personality, personal data, among others. Besides that, the final goal is develop a methodology capable of clustering different types of groups, based on the students' grades, without having to perform any type of pre-processing.

1.3 Motivation and Objectives

In this dissertation, the goal is to solve the problem of assigning students to work groups, recurring to clustering techniques that, given a set of data, allow to create clusters (groups) based on a similarity degree. A methodology will be implemented that will allow to create homogeneous (with a very similar profile) or heterogeneous (with very different profiles) groups, according to what is intended from the activities.

It aims to change the way this is done in classes, where most of the times the students choose their own groups or the groups are chosen randomly, often leading to conflicts. Therefore, it is intended that this different way of creating groups improves not only the learning rate of students in a given class, but also the experience of working with groups different from the ones they are used to.

Ideally, in case of success in this implementation, it can be applied in real situations and impact the way groups are created. Not only this facilitates the job of the teachers but, most important, it promotes collaborative learning and interaction between students, leading to better results in classes.

1.4 Document Structure

Besides the Introduction, this document is composed of three more chapters.

In chapter 2, it is presented a literature review of some approaches that address this matter. These approaches focus on clustering methodologies, but also mention other techniques, such as collaborative filtering and reinforcement learning.

In chapter 3, the problem is described in a more extensive way. A solution proposal is defined, describing the steps of the implementation done. The chapter also discusses the problem of data acquisition and presents a section of data generation, in which there are described different techniques used to generate data. It is concluded by a section explaining how the results are evaluated.

Chapter 4 defines the methodology implemented, describing what algorithms it is based on and detailing all the steps followed.

In chapter 5, there are performed several experiments with the implementation and are presented the results of the groups created. There are also presented some conclusions about the results obtained.

Finally, chapter 6 wraps up the contents of this document, with the final conclusions about this dissertation and the future work to be done.

Chapter 2

Literature Review

2.1 Introduction

Throughout the years, the terms Collaborative Learning (CL) and Learning Management Systems (LMS) have become more and more fashionable. In a brief overview, collaborative learning, within the scope of education, is concerned with the way people can learn together, considering all levels of formal education [SKS06]. Computer-Supported Collaborative Learning is a branch of CL that takes advantage of the advances in computer science and associates them to collaborative learning, allowing students that have different characteristics to cooperate with each other in common tasks [SB15].

This can bring a beneficial effect on learning for the students involved. Therefore, Learning Management Systems (LMS) appear as platforms that enable the set up of a collaborative learning environment, that can be used to gather information about students and create collaborative activities. One of the most known LMS is Moodle. These LMS can make use of external data and/or student activity to create groups that are suitable for the tasks to develop.

This raises the question of finding out what groups can be formed, and with what characteristics and goals. There are two types of groups, homogeneous and heterogeneous. As expected, homogeneous groups have students with similar characteristics while heterogeneous groups are formed of students with a high diversity of characteristics.

There are advantages of using both types. For example, according to [KDHB17], homogeneous groups can be preferred in collaborative learning activities that are suitable for a specific level of knowledge and/or learning style. On the other hand, heterogeneous groups may also be advantageous, because they encourage communication and interaction between students, leading to better results.

However, what characteristics should we consider? This is one of the problems within the task of group formation. Surely there are some attributes that don't have that much relevance when trying to group a person with other individuals. In the context of grouping students, there are some aspects that can be considered. The first one is the academic component. In fact, it is important to know the degree of knowledge and how skilled a student is when trying to assign him to a collaborative learning activity, which can be found by looking at his grades. After that, we can group him with students that have similar skills or students that have completely different capacities, depending on whether we want a homogeneous or heterogeneous group, respectively. This characteristic is the most common used when forming groups within the context of collaborative learning, as it is objective (doesn't depend on anyone's subjectivity) and easily measurable.

However, there are other characteristics that should be considered, specially when the activities require interaction between students. Some of these criteria are:

- communication skills;
- group working attitude;
- thinking styles;
- visual, aural, read/write and kinesthetic (VARK) learning styles, established by [Fle01];
- gender;
- demographic information (there can be a preference for students to work with others that come from the same place as them);
- among others.

As it can be seen from the previous characteristics, although there are some that are measurable, such as gender and demographic information, most of them are very qualitative and depend on the subjectivity of someone to become measurable, hence used in an algorithm.

These qualitative, yet important, characteristics, constitute another problem in the selection of criteria in automatic group formation. In order to solve this problem, we may need the input of the person in charge of the group formation, based on previous interactions, for example. Most importantly, we may need the input of the student themselves, because no one better than them, knows about how they rank themselves in terms of such qualitative and subjective characteristics. One approach is to ask them to fill some questionnaires that then calculate discrete values, based on the answers given (the VARK questionnaire, for example, calculates a score in the end, classifying the student [LSS10]).

This shows that the process of group formation is not entirely independent from students and they should have an input in the task of assigning to the groups, along with the one in charge [Hüb10].

Having all the data available, and decently measured, the next step is to develop a methodology, in order to implement an algorithm that effectively groups the students. There are different techniques that can be used to perform this grouping process. In the next section, there are described some of them, mainly relying on clustering algorithms, and also on multi-agent systems.

Besides these main methodologies, there are other techniques that can be used in order to improve the process of group formation. One example is collaborative filtering.

Collaborative filtering is often related to Recommender Systems, which goal is to "...apply knowledge discovery techniques to the problem of making personalized recommendations for information, products or services during a live interaction." [SKKR01]. In these systems, collaborative filtering works by building a database of preferences of items for the users. So, the database will match a new user with others users that had the same preferences of items.

Transposing this example to the context of group formation, if we consider that the "items" are, for example, the preferences of students, this technique can work as a way of grouping students, thus improving the process.

2.2 Clustering implementations

2.2.1 Clustering

Conceptual clustering is defined by [SM80], as cited in [AMCM83], as "...grouping objects into conceptually simple classes based on attribute values.". Thus, it is expected that the clusters (groups) created have similarities that makes them differentiate from one another. Clustering is adequate for some problems, such as grouping, decision-making and pattern classification, among others [JMF99].

[JD88], as cited in [JMF99] describe five steps in the process of clustering: pattern representation, pattern proximity, grouping, data abstraction and output assessment.

Pattern representation is related to the representation of the data; pattern proximity refers to defining a distance measure function[1] that calculates the distance between two patterns; grouping can be performed according to several algorithms[2]; data abstraction is concerned with the interpretation of what each cluster means and is only performed if needed; output assessment is related to characterizing if the clustering process had good or bad results [JMF99].

Figs. 2.1 and 2.2 depict how clustering is done in a two-dimensional data set. Supposing we need to group the set A, B, C, D, E and F in clusters, if we want to group them in three clusters, the result would be similar to Fig. 2.1. If we wanted, for instance, two groups, the result would be like Fig. 2.2. As it can be seen, the groups are created based on the distance between the data objects, which rely on the distance metrics mentioned above. The following sections describe some of the different methodologies already used in the clustering problem.

[1]Some examples of distance measures are Euclidean, City block, Minkowski or Chebyshev. Details about each and more distance metrics can be found in [Sun07].

[2]The techniques can be divided in hierarchical (such as the agglomerative single-link clustering algorithm or the agglomerative complete-link clustering algorithm) and partitional (squared error clustering method or *k-means* clustering algorithm) approaches [JMF99].

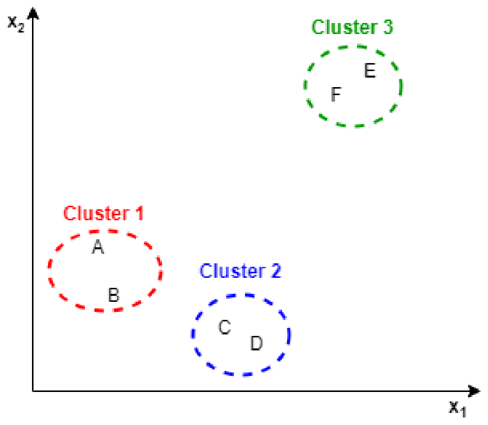


Figure 2.1: Clustering of three groups

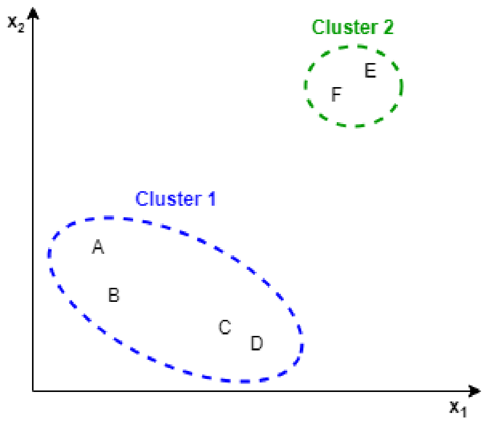


Figure 2.2: Clustering of two groups

2.2.2 *K-Means* Algorithm

K-Means algorithm is a partition clustering algorithm, which goal is to split the data into *k* clusters, each one representing a group. It is one of the most used clustering algorithms, due to the fact that it is "...simple, easy, understandable, scalable and can be adapted to deal with streaming data and very large datasets" [AA13].

The algorithm starts by randomly defining *k* centroids in the data set. Then, the process of partitioning the data follows an objective function. One function widely used is minimizing the square error between the centroids and the data points. This function can be given by [AA13]:

$$E = \sum\sum || p - c_i ||^2 \tag{2.1}$$

,where E is the square error, ci is the centroid and p is a data object.

The algorithm also requires a *distance* function to calculate the distance between the data points and the centroids of the cluster. These functions are the ones mentioned in 2.2.1.

The implementation of this technique can be divided in five steps, as mentioned in [AA13]:

- **1:** specify the *k* number of clusters (usually the number of groups one wants to achieve);
- **2:** the algorithm randomly selects *k* centroids;
- **3:** use of the distance function to assign data objects to a cluster;
- **4:** recompute new centroids based on the points of a cluster;
- **5:** repeat the process until a convergence criterion is reached.

2.2.2.1 Approach to group formation: Spherical K-Means and Expectation Maximization

In [MOW17], based on the premise that group members work better when they are satisfied with their group, the authors proposed an approach to group the students based on some of their characteristics, instead of random grouping. Therefore, the authors made use of the LMS *Moodle* to categorize the students, according to their activity in a *Moodle* discussion forum.

So, the students were classified taking into account three levels of collaboration: independence (students that posted more ideas than replies), interdependence (participation in a discussion, counting the number of responses to questions) and synthesis of information (relation between the comments of a student and the final product). These three elements promote effective CL, according to [MOW17].

In order to perform the grouping, the authors used two objective-function-based machine learning algorithms: Spherical *K-Means* (Skmeans) and Expectation Maximization (EM).

Skmeans is a variant of the *K-means* algorithm. It normalizes the features in order to solve the problem where there are different lengths of features. It then applies then *cosine dissimilarity* [HFKB12] based on the angle between the feature vectors, which can be calculated by:

$$d(p,c_i) = 1 - cos(p,c_i) = 1 - \frac{\langle p, c_i \rangle}{|| \, p \, |||| \, c_i \, ||} \tag{2.2}$$

, being p the data object and ci the centroid. This algorithm is computationally faster than hierarchical clustering when dealing with a large number of variables [MOW17].

Expectation Maximization is an iterative optimization method that has the advantage of estimating data parameters that are missing or hidden [MOW17] [Del02]. Therefore, it finds the maximum likelihood estimates of parameters.

In order to run the clustering algorithms, the authors used the *Weka* software, which is an open source software that allows to apply algorithms related to data mining and machine learning [HFH$^+$09], namely the Skmeans and EM.

In this approach, the goal was to differentiate three types of students, based on skills related to collaborative competence levels, within the discussion forum. Therefore, the students were supposed to be divided in three clusters, representing the levels, which could be high, medium or low.

After retrieving the data from *Moodle* and cleaning it, the authors in [MOW17] fed it to the *Weka* program, using three different data sets. Table 2.1 shows the results of running Skmeans and EM algorithms to the data sets. N is the number of students and C the clusters.

The results show that both algorithms returned similar distribution patterns in the number of students in different clusters. It was also found that the clusters with the lowest values represented students classified as *high* according to the collaborative competence levels, so ranking was required to be done in the clusters, in order to check which clusters represented each level [MOW17].

Table 2.1: Results of clustering with Skmeans and EM. Extracted from [MOW17], adapted.

N	36		109		151	
C	Sk	EM	Sk	EM	Sk	EM
0	9 (25%)	10 (28%)	12 (11%)	61 (56%)	35 (23%)	44 (29%)
1	6 (17%)	18 (50%)	39 (36%)	14 (13%)	35 (23%)	77 (51%)
2	21 (58%)	8 (22%)	58 (53%)	34 (31%)	81 (54%)	30 (20%)

In the end, the clusters created were used to create heterogeneous groups, using an intelligent grouping algorithm.

2.2.2.2 Approach to group formation: AGDP tool

In [SS15], the authors implement a tool called Automated Group Decomposition Program (AGDP), which can be used to form homogeneous, heterogeneous or a mixture of homogeneous and heterogeneous groups of students for the purpose of effective collaborative learning.

Based on the fact that interaction takes a major role in collaborative learning (because people want to defend their viewpoint of subjects, which causes them to have a deeper understanding of the subject), the authors came up with two sets of scores to evaluate the students [SS15]:

- **A-scores:** sum of scores obtained in communication skills, fluency in computer usage and group work attitude;
- **B-scores:** measure of the subject knowledge possessed by each student.

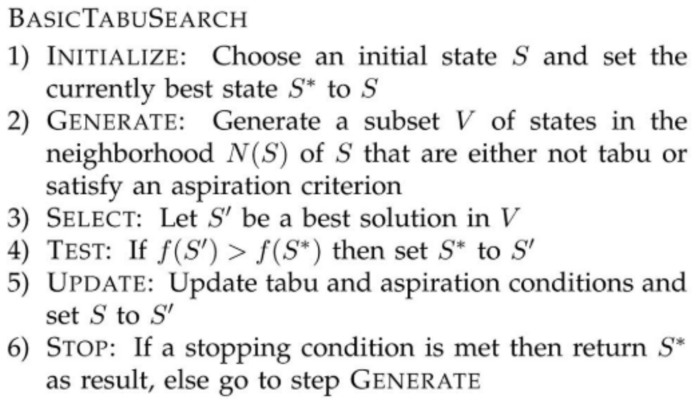
AGDP uses the k-means algorithm and the concept of constraint satisfaction to create groups in an intelligent way. In the study, the constraint to be satisfied is the degree of heterogeneity (DOH), which can be given by (adapted from [SS15]:

$$DOH\ of\ each\ cluster \quad = \quad \frac{\text{Number of ranges of B-scores present in the cluster}}{\text{Cluster size}} \tag{2.3}$$

The implementation of the algorithm is done this way:

- the students are divided into k (defined by the instructor) homogeneous groups, based on the *A-scores*;
- the clusters are equalized to make sure every group as the same amount of students;
- based on the *B-scores*, the clusters are heterogenized according to an implementation that satisfies the DOH.

By comparing the tool with other algorithms, the authors concluded that this method of creating heterogeneous groups reaches a balance in complexity between the more complex heuristic methods and the simpler random selection methods.



```
BASICTABUSEARCH
1) INITIALIZE: Choose an initial state $S$ and set the
   currently best state $S^*$ to $S$
2) GENERATE: Generate a subset $V$ of states in the
   neighborhood $N(S)$ of $S$ that are either not tabu or
   satisfy an aspiration criterion
3) SELECT: Let $S'$ be a best solution in $V$
4) TEST: If $f(S') > f(S^*)$ then set $S^*$ to $S'$
5) UPDATE: Update tabu and aspiration conditions and
   set $S$ to $S'$
6) STOP: If a stopping condition is met then return $S^*$
   as result, else go to step GENERATE
```

Figure 2.3: Basic Tabu Search algorithm. Extracted from [Hüb10]

2.2.3 Tabu Search

As described in [Glo89], "Tabu search is a strategy for solving combinatorial optimization problems...". Although it may seem different from clustering, it is not. In fact, clustering can be seen as an optimization problem: we want the optimize the allocation of objects to groups, following a specific criteria.

However, while cluster analysis try to optimize a certain similarity measure between centroids and the data points, this algorithm's focus is to minimize the distance between two points within a cluster and its center [AS95]. Since Tabu Search is a meta-heuristic method, it is not guaranteed that it finds an optimal solution each time it runs [Hüb10]. Nevertheless, it overcomes other optimization heuristics, such as the Hill Climbing Heuristic, by surpassing the problem of being stuck in a local maximum. It climbs to a local maximum, and then forbids movements that were similar to the ones already made (making them tabu movements). This makes sure that it is not going back to a local maximum, although it requires a systematic use of memory, to keep a history of all the moves made [Hüb10].

Tabu Search algorithm requires that the evaluation function, tabu restrictions and aspiration criteria are defined, additionally to the common elements of heuristic methods. The steps of the algorithm are described in Fig. 2.3.

As it can be seen, the algorithm repeatedly generates states in the neighborhood (that can be reached within one move) of the current state and then moves to the one that maximizes the evaluation function, since it doesn't violate any tabu restriction. To do that, it keeps track of the globally best solution. There can be times where a restriction tries to prevent a good move from being made. In cases like this, the aspiration criteria is used to override the tabu restriction [Hüb10].

2.2.3.1 Approach to group formation: the heuristic Tabu Search algorithm

In [Hüb10], the authors propose the formation of groups based on general criteria, such as academic skills in certain areas, and context-specific criteria, specifically related to the work to develop. In the experiment, the context-specific criteria used are the preferences of the students in working or not with other colleagues, as well as the preferences of the teacher to distribute subsets of students and to assign students to specific groups.

So, a Tabu Search heuristic is used to create maximally diverse groups (heterogeneous) and evenly skilled groups (homogeneous). In the given case, the evaluation function had to consider the skills of the students and the information about the preferences. One good example of how the tabu restrictions and the aspiration criteria impact the algorithm, in this case is: suppose that one of the tabu restrictions is that student A wants to avoid working with student B; however, the next move is to assign student A to a group, which already contains student B. In this situation the move can either be blocked due to the restriction, or it can be overridden by the aspiration criteria, despite of the restriction, if the tabu move leads to a better solution.

By evaluating the computational results of using Tabu Search, the authors in [Hüb10] concluded that running the algorithm took 20-40 seconds (and it is not optimized for speed), which is equivalent to 2 hours work for doing it manually. They also found out that the results were good and the best solution was always optimal with respect to the skills of the students.

Regarding the educational results, the experience was very positive as well. When testing, the evenly skilled group approach was the chosen, because it was the most suitable to the kind of work to develop. In the end, the experience resulted in very few interpersonal problems and positive reactions when the groups were announced.

2.2.4 Multi-agent systems

A multi-agent system is a system composed of several interacting units, the so called *agents*. The system provides an environment, entities and a set of operations that can be applied to the entities, in order to change the environment, being the *agents* the ones who act within the system. Therefore, an *agent* is an entity that understands the environment, acting and communicating with other agents. Agents are expected to have the following properties [Woo09]:

- **reactivity:** as they understand the environment, agents can act upon a change in order to satisfy a given objective;
- **proactiveness:** agents can have a goal-oriented behaviour by taking initiative in satisfying the objective;
- **social ability:** agents can interact with other agents in order to satisfy the objective.

Nowadays, multi-agent systems have their main applications in different scenarios such as problem solving, multi-agent simulation, construction of synthetic worlds, among others.

2.2.4.1 Approach to group formation: Multi-agent group formation

In [KS10], there is implemented a collaborative Wiki called ClassroomWiki, which goal is to track all students' activities and build detailed student models that represent their contributions toward their groups. It also provides a multi-agent framework, the Multi-agent Human Coalition Formation (MHCF) framework, that uses the student models to form students groups in order to improve the collaborative learning of students.

Therefore, the MHCF framework assigns an intelligent agent to each of the participating students, where each agent maintains the model of its assigned student and utilizes that model to estimate the contribution of a student towards his group's Wiki.

The MHCF agent negotiates with others allowing it to solve the current collaborative task well and to improve its knowledge through collaboration to solve future tasks well by forming heterogeneous student groups [KS10].

The formation of groups occurs through a negotiation phase, where a random agent negotiates with other agents in the framework to form a group for its assigned student. The negotiation has three steps: the proposition, where the proposer selects other agents and proposes a group formed by the agents chosen; the consideration, where the proposed-to agent checks the model of the proposer agent and decides if it should join its group or not; the notification phase, in which, if all the proposed-to agents agree to join the proposer's group, the proposer sends a confirmation message to them notifying they now belong to the new group.

To test the system, 145 students were divided in control (groups created randomly) and treatment (groups created using MHCF). Regarding the impact of group formation using MHCF, it was concluded that the treatment set students achieved higher individual scores and also achieved significantly lower standard deviations than the control set students, suggesting that that the collaboration was better in the members of the treatment set.

2.3 Reinforcement Learning

Reinforcement learning (RL) can be described as a "...problem faced by an agent that must learn behavior through trial-and-error interactions with a dynamic environment." [PKLMH96]. Therefore, it works as a way of programming agents by reward and punishment (so that the agents can "learn"), without having to specify the task to solve. The agent has the goal of maximizing the rewards.

It can be achieved by two main strategies: search between the possible behaviors to find the one that performs the best in the environment; using statistical techniques and dynamic programming methods to estimate the utility of taking certain actions. RL can be modeled as a Markov Decision Process, as depicted in Fig. 2.4.

One example for understanding how the agent relates to its environment is depicted in Fig. 2.5. As it can be seen, the agent knows what state it is in and what options it has. Then it chooses an action and receives a reward or punishment, moving to the next state. So, there is no way of

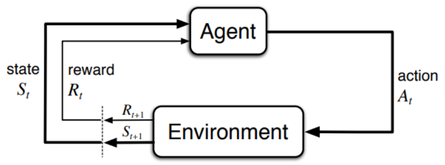


Figure 2.4: Markov Decision Process. Extracted from [SB98]

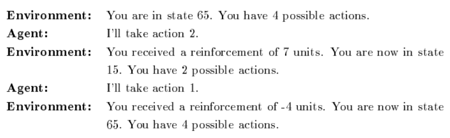

Environment: You are in state 65. You have 4 possible actions.
Agent: I'll take action 2.
Environment: You received a reinforcement of 7 units. You are now in state 15. You have 2 possible actions.
Agent: I'll take action 1.
Environment: You received a reinforcement of -4 units. You are now in state 65. You have 4 possible actions.

Figure 2.5: Example of agent/environment interactions. Extracted from [PKLMH96]

knowing which action would be better in the long-term. Instead, it learns by trial and error. This is one of things that differ RL from other learning paradigms, such as supervised learning. Another difference is that the evaluation of the systems occurs concurrently with learning.

RL algorithms are already used to simulate human behavior in several situations. For example, in [MKS+15], an intelligent agent is developed, recurring to the training of deep neural networks, which is able to achieve the same performance of a professional games tester across 49 different games, using the same algorithm, network architecture and hyper-parameters. Besides applications in game theory, there are others, such as in information theory and operations research. However there aren't implementations directly related to group formation.

2.4 Conclusions

In a brief conclusion about all the methodologies, algorithms, techniques and implementations mentioned below, it can be said that most of them already are directly related to the problem of automatic group formation. However, most of the implementations rely on the use of objective and academic related criteria (grades, participation in forums, thinking and learning styles), and lack in some attributes, like students preferences and some of their personal data.

Finally, it can also be noticed that the the there isn't often a combination of methodologies and techniques, which could improve the process of group formation.

Chapter 3

Automatic Group Formation

This dissertation aims to provide a solution to the task of grouping people. More specifically, the main focus is the automatic formation of groups of students in classes. This process is relevant, mainly due to the importance given to concepts like Collaborative Learning, Learning Management Systems and Computer-Supported Collaborative Learning.

In this chapter, we describe the problem we want to solve, along with the solution proposed. We also approach the problem of data acquisition, that arised with the development of the project. It is followed by a section in which we explain how we overcame this issue, presenting some methodologies that were developed and used to solve this problem. The chapter is finalized by a section explaining how the results were evaluated.

3.1 Problem definition

Nowadays, collaborative learning is an emerging concept in the field of Education. It is believed that, besides the tradicional way of learning (individually, from a teacher), students can also learn from each other. Taking this into account, the way the students are grouped is important to ensure that the groups are well balanced, so that all students can benefit from the experience of working with each other.

In this dissertation, the problem we want to solve is the one of creating groups of students, based on specific criteria. It is intended to create two types of groups: homogeneous (groups where the students have similar characteristics) and heterogeneous (groups where the characteristics of the students diverge) groups. These types are defined by the user and are chosen according to the activity to develop. This grouping task is pretended to be done by means of clustering techniques.

In order to create such groups, it is important to define what criteria to use. The main attribute to consider is the academic component, the grades of the students, since it is what characterizes them the most. However, there are other interesting attributes that can be considered.

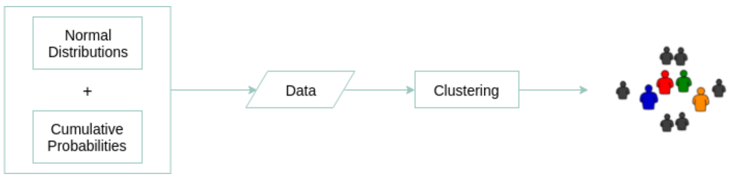


Figure 3.1: Model of the proposed solution

One of them is the personal data of the students, like their location. Students may work better with people that come from the same place, either for cultural reasons or simply because they know each other.

Another interesting feature can be looking at the preferences of the students, because they can prefer working with some people and avoid working with other. This makes the task of grouping more complete, since the decision isn't left intirely to the teacher, and students feel included in the process.

In this project we propose a solution that only uses the grades of the students as input for clustering. This obviously doesn't discard the importance of the other attributes mencioned. In fact, they should be used in future implementations, to add a more personal component to the students' profiles.

3.2 Proposed Solution

The solution proposed in this dissertation has similar goals to the implementations proposed in Section 2. However, it aims to improve the implementations that already exist by adding more options to the clustering process, like the ability of clustering by the differences of students, with the goal of providing a wider range of choices to the user in charge of the grouping task.

Therefore, the main objective of this dissertation is to automatically form groups of students recurring to clustering algorithms. In addition to that, the solution includes a section of data generation, in which are used some methods to create data, with an approximation to real situations.

To achieve that, techniques like normal distributions and cumulative probabilities are used, as we will explain in the following sections. Fig. 3.1 depicts the model for the proposed solution.

The implementation can be described in the following four steps:

- **phase 1:** understand what attributes can be used to group students;
- **phase 2:** generate data using normal distributions and cumulative probabilities;
- **phase 3:** implement a methodology, based on clustering techniques, and test it with the data;
- **phase 4:** evaluate the results.

3.3 Data Acquisition

According to [ABRA+18], there are three main data sources: existing databases with real data, data simulation and empirical data. Originally, the goal was to use real data from FEUP students. This would make easier the process of evaluating the results, that could be done by testing the groups formed by the students involved and checking the results. However, due to problems related to data protection, acquiring such data was not possible.

So, another option would be to use real data from available data sets related to students information. There are data sets online that fit this purpose, like [Stu]. However, most of them are unavailable. Also, they often don't have all the information required for this implementation. For example, they tend to be very complete in terms of grades and activity of the students, but they are poor in terms of personal data and preferences of students.

Based on that, the data acquisition method that will be used in this dissertation is data simulation, since it is one of the most common approaches and we don't need to depend on third parties to have complete data. As expected, the data will not be randomly simulated. By doing that, one could get data that does not simulate well a real environment. For example, if one predicted that all grades from all students were 0, this would heavily impact the groups created.

Instead, the goal is to make use of distributions that try to approximate the most the data simulated to real situations. By doing that, we can make sure the results can be adapted to real scenarios.

3.4 Results evaluation

In terms of evaluating the results, several experiments were conducted, using different data generation methods, datasets and number of groups.

Also, we calculated some statistics that allowed to define the profile of the clusters. These statistics include the means, standard deviation and homogeneity of the resulting clusters, providing useful information for the analysis of the results.

3.5 Data Generation

As mentioned in Chapter 3, the data used in the experiments was acquired through data simulation. In this chapter there are presented the techniques used to overcome the problem of generating the data required. The data generated are the course units' grades, for each student. The data simulation was performed through three aproaches presented in the following sections: fixed values, using normal distributions, and using normal distributions and cumulative probabilities.

3.5.1 Fixed values

The first approach, and most simple, was to *hard code* values in order to create Low, Medium and High tier students (where the Low tier represents students with lower grades and the High tier represents the opposite). An example of this approach can be found in Table 3.1, where students are identified in rows, unit courses in columns, and grades are the cells' values between 0 and 20.

Table 3.1: Hard coded students' grades

	UC1	UC2	UC3	UC4	UC5	UC6
A	10	10	10	10	10	10
B	10	10	10	10	10	10
C	10	10	10	10	10	10
D	14	14	14	14	14	14
E	14	14	14	14	14	14
F	14	14	14	14	14	14
G	16	16	16	16	16	16
H	16	16	16	16	16	16
I	16	16	16	16	16	16
J	20	20	20	20	20	20
K	20	20	20	20	20	20
L	20	20	20	20	20	20

The data created through this method, besides not being accurate nor representing a good simulation of reality, plays an important role in the interpretation of the results obtained, after the clustering. The clear differences in the groups created allow us to take some conclusions about the methodology implemented, since data can be generate in a way to demonstrate typical or extreme situations. This is shown in more detail in the following chapters.

So, the approaches presented next rely on a Normal Distribution to define the academic profiles of each student.

3.5.2 Using normal distribution

In the next attempts, we tried to use normal distributions of some attributes in order to obtain realistic data.

In a brief explanation, the normal distribution (or Gaussian distribution) is a probability function, describing how the values of some attribute are distributed. This distribution can be represented by a bell curve, as depicted in Fig. 3.2.

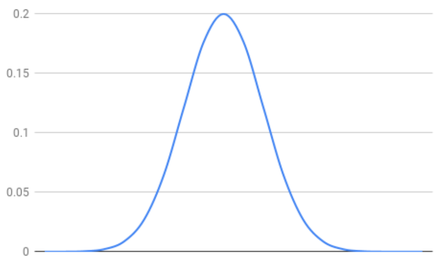


Figure 3.2: Example of a Bell Curve

The normal distribution has some parameters that define the shape of the curve. The main parameter is the *mean*, that defines the peak of the curve. The values of the distribution have higher probability of being close to the mean value. Other important parameter is the *standard deviation*. It defines how wide is the normal distribution and how far can the values diverge from the mean.

Applying this concept to the problem in question, the *mean* parameter represents the mean of the course units' grades, in some cases, or the mean of the students' grades, in others, as presented next.

3.5.2.1 Mean of the course units

In the first attempt using a normal distribution, the goal was to generate grades for 12 students, based on 6 course units[1]. Therefore, the values for the *mean* and *standard deviation* were arbitrary chosen. An example is shown in Table 3.2.

Table 3.2: Example of course units means and standard deviations

	Mean	**Standard Deviation**
UC1	12	2
UC2	13	2
UC3	15	2
UC4	17	2
UC5	16	2
UC6	14	2

We opted to vary the values of the mean of each course unit from a range from 12 and 17, in an attempt to simulate easier and harder course units, respectively. The standard deviation used was 2 for each course unit.

When generating the distribution for the 12 students, the grades of each student were restricted to a range of values from 7 to 20 (it is unrealistic to have students with grades lower than 7), although the probability of obtaining these border values is low, according to the definition of the normal distribution. An example of the results can be shown in Table 3.3.

However, this implementation presents a major setback. Since the grades of each course unit are independent from each other, there can be a case where a student has very high grades in some course units and very low grades in others. An example can be ssen in Table 3.4.

Student *X* represents an unrealistic case, because students often have similar grades in the majority of the course units, and the values don't diverge to extremes. This issue can be solved by considering also the mean of students' grades, as described next.

[1]The values 12 and 6 are for illustration purposes only. The methodology implemented allows to use any number of students and course units.

Table 3.3: Grades generated according to the means of each course unit

	UC1	**UC2**	**UC3**	**UC4**	**UC5**	**UC6**
A	14	13	17	19	18	14
B	12	12	16	18	14	17
C	10	15	15	20	15	12
D	8	13	12	20	18	15
E	17	16	16	15	13	16
F	16	13	17	14	16	15
G	13	15	16	17	14	15
H	15	13	14	16	13	17
I	12	14	12	15	17	11
J	9	14	18	16	17	14
K	13	11	14	14	16	13
L	16	10	15	11	16	13

Table 3.4: Possible grades of a student

	UC1	**UC2**	**UC3**	**UC4**	**UC5**	**UC6**
X	10	9	11	17	18	19

3.5.2.2 Mean of the students' grades

In the following approach, we aimed to improve the quality of the data generated by attempting to solve the problem of extreme values in the grades. In order to do that, we decided to create a normal distribution of the values of the means of the students.

Using the same set of students as in the previous experiment, we calculated a a normal distribution of the means of the student, using the parameters *mean = 14* and *standard deviation = 2*.

After that, for each student we calculated a new distribution, using the mean of the student as the mean parameter and keeping the standard deviation with the value of 2. This distribution was used to calculate the grades of each course unit, for each student. The final results are presented in Table 3.5.

Table 3.5: Grades generated according to the means of each student

	Mean	**UC1**	**UC2**	**UC3**	**UC4**	**UC5**	**UC6**
A	17	15	15	15	14	16	17
B	13	12	11	17	16	14	14
C	17	14	10	13	15	14	15
D	20	12	13	17	18	18	12
E	14	14	12	16	16	15	15
F	13	12	11	15	14	16	15
G	19	16	17	18	14	16	14
H	14	12	13	16	11	15	18
I	13	13	18	13	12	16	16
J	18	16	14	15	11	16	16
K	14	14	19	16	14	15	17
L	17	13	12	17	14	13	18

This experiment allowed us to obtain a more accurate generation of results, eliminating unrealistic cases, thus solving the problem of the previous solution.

3.5.3 Using normal distributions and cumulative probabilities

Although the previous solution presents plausible results, we decided to take the data generation problem one step further, by combining the normal distribution of the course units' grades and the means of the students at the same time. An algorithm was developed and it works as follows:

Algorithm 1: Generate Population

```
Input: $nStudents, mean, nUcs$
GenPopulation ($nStudents, mean, nUcs$)
    $population := []$;
    $names := generateNames(nStudents)$;
    $means := generateMeans(nStudents, mean)$;
    $ucsMeans := generateUcsMeans(nUcs, mean)$;
    foreach $mean\ m_i \in means$ do
        $grades := generateRandomGrades(m_i)$;
        while $not\ allGradesAssignedToUcs$ do
            $cumulProb := generateCumulativeProbabilities(ucsMeans)$
            $g := getHighestGradeNotAssigned(grades)$;
            $rand := generateRandomNumber01()$;
            $ucAssigned := assignGradeToUc(g, rand)$;
            $removeUcAssigned(ucAssigned)$;
            $removeGrade(g)$;
        end
        $student := appendNametoGrades(grades, names)$;
        $appendStudentToPopulation(student)$;
    end
    return $population$;
```

Algorithm 1 has 3 input parameters: *n_students*, the number of students to generate the grades for; *mean*, the mean value of the students' means; *n_ucs*, the number of course units to take into account.

In the first step of the algorithm, it generates names for the students, represented by capital letters (e.g. *A, B, C, ...*), as well as the means of each student. When generating the means, the data created follows a normal distribution, using the *mean* parameter as input of the distribution. There are defined the names of the course units, represented by the prefix "UC", followed by a number (e.g. *UC1, UC2, ...*). Also, the list of course unit grades is created, using a normal distribution with the *mean* parameter, and narrowing the values from 12 to 17. The reason for this is that it is not plausible that a course unit has a mean of very high or very low values.

The second step is a loop that generates the grades of each student. Therefore, using the mean of the student in question, a list of grades (with size equal to the *n_ucs* parameter) is created, following a normal distribution. The list is then sorted by ascending order.

The third and final step is to assign each grade of the student to a specific course unit. So, in first place, a list with the cumulative probabilities of each course unit is created (Table 3.6).

Table 3.6: Cumulative probabilities of the course units

	Mean	P(UCi)	Cum P(UCi)
UC1	12	0.1363636364	0.1363636364
UC2	13	0.1477272727	0.2840909091
UC3	15	0.1704545455	0.4545454545
UC4	15	0.1704545455	0.625
UC5	16	0.1818181818	0.8068181818
UC6	17	0.1931818182	1

Then, using the highest grade of the students' grades, it is assigned to a specific course unit, using a random function that generates a value between 0 and 1. Both the highest grade and the course unit it is assigned to are removed from the equation and the process is repeated until all grades are assigned to a course unit.

This is done for all students. An example of the final results is shown in Table 3.7, for twelve students and six unit courses.

Table 3.7: Grades generated using normal distribution and cumulative probabilities

	UC1	UC2	UC3	UC4	UC5	UC6
A	11	12	12	12	12	12
B	14	12	13	12	14	12
C	11	14	13	15	14	14
D	15	16	16	17	17	18
E	13	12	15	14	12	14
F	12	12	12	12	12	11
G	18	16	16	17	16	16
H	13	15	14	14	14	15
I	11	13	12	13	12	11
J	11	14	13	15	12	12
K	12	13	12	12	13	12
L	12	11	11	10	11	12

Chapter 4

Implementation

As explained before in this dissertation, we want to solve the problem of assigning students to groups, according to a specific criteria. Also, we aim to solve this problem using clustering techniques. After looking to some of the existent implementations and techniques used to approach this topic (presented in chapter 2), we decided to use an adaptation of the *k-medoids* algorithm, which is a variation of the *k-means* algorithm.

4.1 *K-medoids*

The *k-medoids* algorithm works in a very similar way of the *k-means* algorithm, presented in 2.2.2.

One difference and advantage of *k-medoids* is in the step of selecting the centroid. In *k-means*, the centroids are space points, representing the mean of the data points of each cluster. This presents a drawback because, in case of extreme data points, the recalculation of the centroids can result in improper clustering [Bha14]. Thus, *k-means* is sensitive to outliers.

K-medoids solves this problem because the centroids (medoids) chosen are the data points that minimize the dissimilarity of the cluster. The dissimilarity is the difference between the point labeled as the center of the cluster and the other points assigned to that cluster.

The impact of this difference in both algorithms is depicted in Fig. 4.1. In the figure we can check that the rightmost data point is an outlier of the cluster. When calculating the centroid in *k-means* the outlier will impact the mean of the cluster. However, when calculating the medoid in *k-medoids*, since the medoid is a data point, it will be closer to all the others than to the outlier.

When trying to recompute the medoids, the *k-medoids* algorithm tries to find which point of each cluster minimizes the intra-cluster dissimilarities. In order to do that, the algorithm swaps each non-medoid point of the cluster with the medoid, trying to find the point that minimizes the dissimilarity. This point is then used as medoid for the next iteration of the algorithm. When none of the medoids change, it means the distance intra-cluster is minimal, and the algorithm as reached a solution.

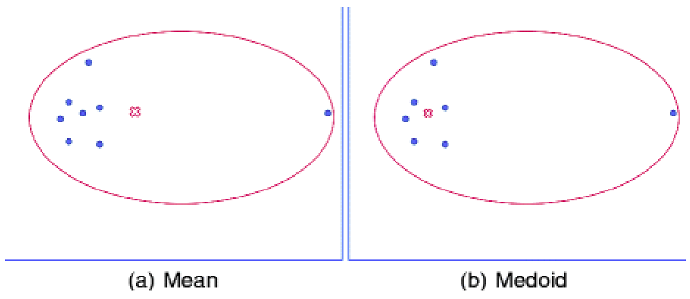


Figure 4.1: Selection of a mean and a medoid. Extracted from [JH10]

Similarly to *k-means*, *k-medoids* also uses the same distance functions to calculate the distance between points.

4.2 Adapted *k-medoids*

In our implementation, we use an adaptation of the *k-medoids* algorithm. It differs in some aspects, because it is directed to the problem of creating groups of students.

So, the first difference is that we want to create groups of students of the same size and none of the clustering techniques presented satisfies this restriction. Both *k-means* and *k-medoids* return clusters of different sizes, depending on the profile of the data points.

To solve this problem, we added a restriction in the function used to assign a point to a cluster. In first place, the cluster is characterized as *full* if the number of data points assigned to it is equal to the total number of data points divided by k. Then, when assigning a new point to a cluster, if it isn't full, the point is added to that cluster. Otherwise, the algorithm checks if there is any point of the cluster that has a distance to the centroid higher than the distance from the new point to the centroid. If this happens, the point in the centroid is removed and replaced by the new point and it has to be assigned to a new centroid. This is done until all points are assigned to a cluster. Algorithm 2 explains this process.

The second difference is related to the distance function used. The most common used are the *Euclidean* and *Manhattan* distance metrics. However, in our case, a simple distance metric isn't enough, since we want to be able to create homogeneous and heterogeneous groups. Also, another component added to our algorithm is the fact that the user can give a weight to each of the unit courses. This was done to give the ability to chose which course units have more or less impact in the groups formed. It is useful because there are situations where some unit courses are more related to the course unit in which we want to create the groups. Exemplifying, let's suppose we want to group students of a class in a course unit related to the subject of Artificial Intelligence. Assuming we have information about the grades of 3 unit course grades of the students and that the subjects of those course units were Machine Learning, Artificial Intelligence and Data Mining, it makes sense that the unit course grades of the course unit with the subject of Artificial Intelligence should impact the outcome the most.

Algorithm 2: Assign point to cluster

```
Input: $point, centroids, clusters$
AssignoToCluster ($point, centroids, clusters$)
    $clusterToAssign := getClosestCluster(point, centroids)$;
    if clusterToAssign is full then
        if found newPoint with higher distance then
            $addToCluster(point, clusterToAssign)$;
            $assignToAnotherCluster(newPoint)$;
        end
        else
            $assignToNextClosestCluster(point)$;
        end
    end
    else
        $addToCluster(point, clusterToAssign)$;
    end
```

Therefore, in case of homogeneous groups, we used a variation of one of the most common metrics. It was based on the *Euclidean* distance function, in which the distance between two points is given by:

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{4.1}$$

, where p and q are the data points and n is the number of attributes of the points.

Adding the weights to each class, the distance metric we used is defined by:

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2 * weight_i} \tag{4.2}$$

In case of heterogeneous groups, there isn't any metric function that groups points by the differences between them. So, we defined a new distance function with the purpose of solving this problem. It is given by:

$$d(p,q) = MAX\_GRADE - \sum_{i=1}^{n}(\mid p_i - q_i \mid * weight_i) \tag{4.3}$$

, where d and p are data points, n is a the number of attributes of the points and *MAX_GRADE* is the maximum grade a student can have (in our experiments, the value used was *20*).

Through this equation, we can minimize the distances between points that are very different. Table 4.1 shows an example of some points and the distance between them, applying the equation presented in equation 4.3 and using the same weight to each unit course. As it can be seen, the

Table 4.1: Distances between different points in heterogeneous groups

	ATTR_1	ATTR_2	ATTR_3
A	20	20	20
B	18	18	18
C	2	2	2

DIST	A	B	C
A	-	18	2
B	18	-	4
C	2	4	-

distance between points *A* and *B* (*d(A,B) = 18*) is much greater than the distance between *A* and *C* (*d(A,C) = 2*), although *A* is more similar to *B* than *C*.

4.3 Algorithm implemented

After solving these two issues, we came up with Algorithm 3.

Algorithm 3: Adapted *K-Medoids* pseudocode

```
Input: point, centroids, clusters
AdaptedKMedoids (data, k, metric)
    centroids := choseRandomCentroids(data, k);
    bestSolution := false;
    while not bestSolution do
        bestSolution := true;
        clusters := calcClusters(data, centroids, metric);
        cost := calcTotalCost(clusters, centroids, metric);
        checked := [];
        while checked.length() < data.length() - k do
            newCentroids := calcNewCentroids(Data, checked, centroids, clusters);
            newClusters := calcClusters(data, newCentroids, metric);
            newCost := calcCost(newClusters, newCentroids, metric);
            if newCost < cost then
                bestSolution := false;
                centroids := newCentroids;
                cost := newCost;
                checked := [];
            end
            else
                checkd.append(newCentroids);
            end
        end
    end
    return clusters, cost;
```

The algorithm has three input parameters, the *data* we want to group, *k*, representing the number of groups to create, and the *metric*, which defines if the groups formed should be homogeneous or heterogeneous (defined by the integers 0 and 1, respectively).

First, there are arbitrated (in the method *choseRandomCentroids()*) *k* data points to be the initial centroids of the clusters. Then, the algorithm begins a *While* cycle, where it initially assumes it has the best solution. In the cycle, the clusters are calculated, using the centroids defined earlier, along with the associated cost of the clusters formed. The total cost of the clusters is the sum of the intra-cluster distances (described in equations 4.2 and 4.3, according to the *metric* defined).

Next, an array *checked* is created. This array is responsible for storing the centroids that already were verified.

After that the algorithm enters a new loop, that tries to replace the current centroids with other data points and check if the total cost of the clusters created decreases. If that happens, the points selected become the new centroids, and the cycle repeats the process. If not, the non-centroid points are added to the checked list. If, after trying all thenon-centroid data points, the total cost of the clusters doesn't decrease, it means the algorithm has reached the best solution, thus returning the clusters created and the associated cost.

This implementation follows the principles of the *K-medoids* algorithm. Because of that, it may present a problem, because *k-medoids* converges to a local optimum solution, meaning that the final result depends on the data points selected as initial centroids. If the algorithm choses initial centroids that decrease the most the intra-cluster cost it may not change and return that solution. However, this doesn't mean we couldn't obtain a better solution by swapping some points between different clusters.

Therefore, it is possible that the final solution isn't necessarily the best overall solution. To solve this probem, what we did was to run the algorithm several times, with different starting points. After adding this, the final algorithm is defined in Algorithm 4. Although this solution doesn't guarantee the best solution, it highly decreases the probability of the algorithm being stuck in a local optimum.

Algorithm 4: Group students pseudocode

```
Input: $point, centroids, clusters$
GroupStudents $(data, k, metric)$
    $cost := MAX\_INT$;
    $clusters := []$;
    for $i = 0$ to $10$ do
        $new_clusters, new_cost := AdaptedKMedoids(data, k, metric)$;
        if $newCost < cost$ then
            $clusters := newClusters$;
            $cost := newCost$;
        end
    end
    return clusters, cost;
```

The implementation of all the pseudocode presented before was done using the Python programming language, for its ease with working with lists and arrays. All the code is presented in the Appendix.

Table 4.2: Example of output file with the results

	UC1	**UC2**	**UC3**	**CLUSTER**
A	10	20	20	cluster_0
B	20	20	20	cluster_0
C	16	16	16	cluster_0
D	13	13	13	cluster_0
E	10	10	10	cluster_1
F	10	10	10	cluster_1
G	2	2	2	cluster_1
H	2	2	2	cluster_1

Also, a simple command line interface was developed. Before calling the *GroupStudents* method, the user has the ability of inserting a *csv* file with the data he wants to group or to use automatic generation of data, described in section 3.5. He is also presented an option to define weights for each course unit or to give the same weight to each course unit. In the end, the program presents him the results in a file called *groupedStudents.xlsx* which contains the attributes of the students and the group they were assigned to. An example can be seen in Table 4.2.

In the next chapter we will present some experiments done with this methodology and make an analysis of the results obtained.

Chapter 5

Experiments and Evaluation

In order to conclude about the methodology implemented, some experiments were conducted using the data generated in 3.5. So, in this chapter we will present what data was used and the final results obtained with it. We will start by presenting results obtained with fixed and distinct values, followed by results of experiments with data generated through normal distributions and cumulative probabilities.

5.1 Experiments with fixed values

The purpose of this first experiment was to easily see how the process should work and how the methodology behaves. The data used clearly divides the students in several levels and it is easy to analyze the groups created and see that the algorithm worked as it should. This data is shown in Table 5.1.

Table 5.1: Dataset of 12 students using fixed values

	UC1	**UC2**	**UC3**
A	20	20	20
B	20	20	20
C	20	20	20
D	20	20	20
E	16	16	16
F	16	16	16
G	16	16	16
H	16	16	16
I	10	10	10
J	10	10	10
K	10	10	10
L	10	10	10

Using this dataset, let's suppose we wanted to group the students in homogeneous groups of 4 students. Looking at the table, we can easily group the students by their similar grades, putting students *A, B, C* and *D* in a group, *E, F, G* and *H* in another and finally *I, J, K* and *L*. Similarly, if we wanted to create groups of the 3 students, but heterogeneous, the groups would be *[A, E, I], [B, F, J], [C, G, K], [D, G, L]*, or variations of it (the restriction being that students with the same grade can't be in the same group).

As expected, the algorithm also returns the same results. We put this data in a *csv* file and then fed it to the algorithm. In the first experiment, we used parameters *k = 3* and *metric = 0* (homogeneous groups). In the second experiment, we changed the parameters to be *k = 4* and *metric = 1* (heterogeneous groups). The results of both experiments are presented in Tables 5.2 and 5.3, respectively.

Table 5.2: Homogeneous groups, *k=3*

	UC1	UC2	UC3	CLUSTER
A	20	20	20	cluster_1
B	20	20	20	cluster_1
C	20	20	20	cluster_1
D	20	20	20	cluster_1
E	16	16	16	cluster_2
F	16	16	16	cluster_2
G	16	16	16	cluster_2
H	16	16	16	cluster_2
I	10	10	10	cluster_0
J	10	10	10	cluster_0
K	10	10	10	cluster_0
L	10	10	10	cluster_0

Table 5.3: Heterogeneous groups, *k=3*

	UC1	UC2	UC3	CLUSTER
A	20	20	20	cluster_0
B	20	20	20	cluster_3
C	20	20	20	cluster_1
D	20	20	20	cluster_2
E	16	16	16	cluster_0
F	16	16	16	cluster_2
G	16	16	16	cluster_3
H	16	16	16	cluster_1
I	10	10	10	cluster_3
J	10	10	10	cluster_0
K	10	10	10	cluster_1
L	10	10	10	cluster_2

Although the result of the second experiment isn't exactly like the one predicted in the begining, it follows the same principle, which was to assign students with the same grades to different groups. As stated before, this experiment is for demonstration only, since the values of the grades used don't represent real situations.

5.2 Experiments using normal distributions and cumulative probabilities

In the following experiments, we will present the results obtained using the generation of data shown in Sections 3.5.2 and 3.5.3. In all the experiments, the attribute used for the *mean* parameter of the distribution was 14. The first experiment was based on a dataset of 12 students and the second uses a dataset of 24 students, with the objective of representing a real class environment.

5.2.1 Experiment 1 - 12 students

In this experiment, we tried to group 12 students in homogeneous and heterogeneous groups. We chose a value of *k = 4*, and then compared the mean grade of each cluster created by the algorithm. The dataset used was generated by the method described in 3.5.2. It can be consulted in Table 5.4.

Table 5.4: Dataset of 12 students created with normal distribution

	UC1	**UC2**	**UC3**	**UC4**	**UC5**	**UC6**
A	15	15	15	14	16	17
B	12	11	17	16	14	14
C	14	10	13	15	14	15
D	12	13	17	18	18	12
E	14	12	16	16	15	15
F	12	11	15	14	16	15
G	16	17	18	14	16	14
H	12	13	16	11	15	18
I	13	18	13	12	16	16
J	16	14	15	11	16	16
K	14	19	16	14	15	17
L	13	12	17	14	13	18

The results of running the algorithm to form both types of groups are shown in Tables 5.5 and 5.6.

Table 5.5: Results of homogenous groups

	Homogeneous Groups											
Clusters	**Cluster 0**			**Cluster 1**			**Cluster 2**			**Cluster 3**		
Names	**C**	**E**	**F**	**A**	**H**	**J**	**B**	**D**	**L**	**G**	**I**	**K**
UC1	14	14	12	15	12	16	12	12	13	16	13	14
UC2	10	12	11	15	13	14	11	13	12	17	18	19
UC3	13	16	15	15	16	15	17	17	17	18	13	16
UC4	15	16	14	14	11	11	16	18	14	14	12	14
UC5	14	15	16	16	15	16	14	18	13	16	16	15
UC6	15	15	15	17	18	16	14	12	18	14	16	17

Table 5.6: Results of heterogeneous groups

	Heterogeneous Groups											
Clusters	**Cluster 0**			**Cluster 1**			**Cluster 2**			**Cluster 3**		
Names	**E**	**F**	**K**	**A**	**D**	**J**	**B**	**I**	**L**	**C**	**G**	**H**
UC1	14	12	14	15	12	16	12	13	13	14	16	12
UC2	12	11	19	15	13	14	11	18	12	10	17	13
UC3	16	15	16	15	17	15	17	13	17	13	18	16
UC4	16	14	14	14	18	11	16	12	14	15	14	11
UC5	15	16	15	16	18	16	14	16	13	14	16	15
UC6	15	15	17	17	12	16	14	16	18	15	14	18

For better analysis of the groups created, we built charts with the means of the grades of the students, for each cluster. They are presented in Figs. 5.1 and 5.2.

The results are very inconclusive about whether or not the algorithm performed well. As it can be seen by the figures, there is not much difference in terms of the means of the clusters created. Despite the fact that the students' grades were generated through a normal distribution, their means don't vary that much. This could be related to two factors: the first being that the number of data

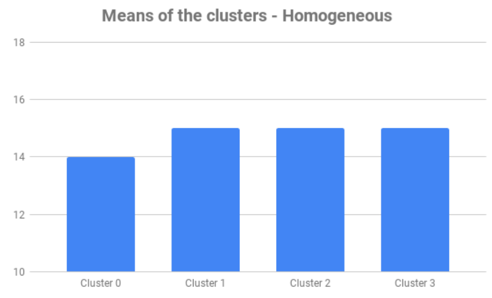


Figure 5.1: Means of the homogeneous clusters

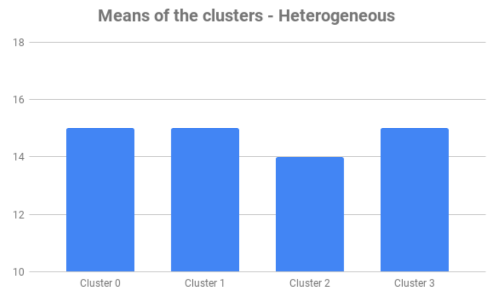


Figure 5.2: Means of the heterogeneous clusters

generated was small (only 12 students); the second related to the *standard deviation* used, which didn't allow the means to be different from each other.

In order to find the source of this problem, we generated new data, but following the algorithm developed in section 3.5.3. The dataset generated is shown in Table 5.7.

Table 5.7: Dataset of 12 students created with normal distribution and cumulative probabilities

	UC1	**UC2**	**UC3**	**UC4**	**UC5**	**UC6**
A	13	14	14	13	13	12
B	16	14	15	16	17	14
C	8	9	9	9	11	8
D	9	10	9	11	10	10
E	13	11	10	12	13	13
F	18	18	17	18	18	17
G	12	12	14	12	15	13
H	15	14	13	13	14	15
I	13	12	13	12	14	12
J	14	15	14	15	14	13
K	15	14	15	15	14	15
L	14	12	16	14	15	16

Running the same methodology implemented in this data, the results were different, as it can be seen in Tables 5.8 and 5.9 and Figs. 5.3 and 5.4.

Table 5.8: Results of homogenous groups

	Homogeneous Groups											
Clusters	**Cluster 0**			**Cluster 1**			**Cluster 2**			**Cluster 3**		
Names	**C**	**D**	**E**	**B**	**F**	**L**	**A**	**G**	**I**	**H**	**J**	**K**
UC1	8	9	13	16	18	14	13	12	13	15	14	15
UC2	9	10	11	14	18	12	14	12	12	14	15	14
UC3	9	9	10	15	17	16	14	14	13	13	14	15
UC4	9	11	12	16	18	14	13	12	12	13	15	15
UC5	11	10	13	17	18	15	13	15	14	14	14	14
UC6	8	10	13	14	17	16	12	13	12	15	13	15

Through this method of data generation, we can easily see the differences in the clusters created. When creating homogeneous groups, the mean of the clusters is very different. We can

Table 5.9: Results of heterogeneous groups

	Homogeneous Groups											
Clusters	**Cluster 0**			**Cluster 1**			**Cluster 2**			**Cluster 3**		
Names	**E**	**F**	**I**	**A**	**B**	**G**	**C**	**K**	**L**	**D**	**H**	**J**
UC1	13	18	13	13	16	12	8	15	14	9	15	14
UC2	11	18	12	14	14	12	9	14	12	10	14	15
UC3	10	17	13	14	15	14	9	15	16	9	13	14
UC4	12	18	12	13	16	12	9	15	14	11	13	15
UC5	13	18	14	13	17	15	11	14	15	10	14	14
UC6	13	17	12	12	14	13	8	15	16	10	15	13

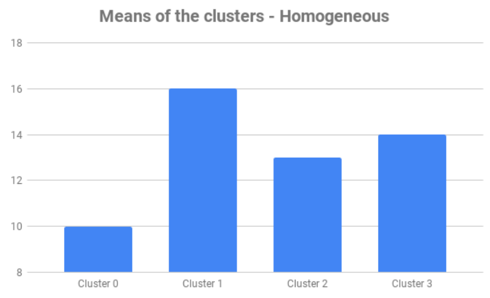


Figure 5.3: Means of the homogeneous clusters

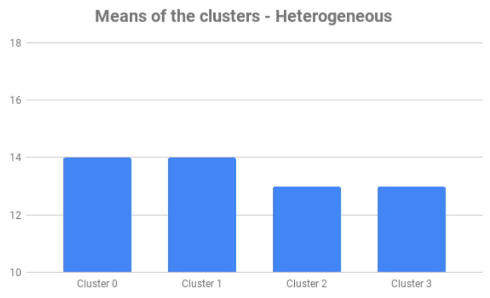


Figure 5.4: Means of the heterogeneous clusters

obviously see that the students were divided by tiers. In the example of Fig. 5.3, the values diverge from a mean of 10 to 16. However, when we try to cluster the same set of students in an heterogeneous way, the groups are much more balanced, only diverging from a mean of 13 to 14.

5.2.2 Experiment 2 - 24 students

The previous experiments allowed us to see how the data can impact the outcome of the methodology developed. However, the purpose of this experiment was to perform clustering in a dataset that could represent a real life environment. Because of that, we used a dataset composed by 24 students that was generated by the algorithm developed in section 3.5.3. It can be consulted in Appendix A, in A.1, as well as the results of the experiments conducted.

Several tests were conducted to simulate real class situations where the students have to group themselves. The first test simulates a situation where the students have to work in pairs. Applying our algorithm to this situation, we get the results presented in Figs. A.1 and A.2.

In the second experiment, the goal was to create 4 groups of 6 students each. We finally added a third experiment, where we grouped the 24 students in 6 groups of 4 students. The results are depicted in Figs. A.3, A.4, A.7 and A.8, for both tests.

For the last 2 experiments, we also calculated the standard deviation of the grades inside each cluster. For the last experiment, Fig. 5.5 shows the standard deviation in homogeneous groups and Fig. 5.6 in heterogeneous groups.

The graphics clearly indicate that the standard deviation in homogeneous groups reaches much lower values than the standard deviation in heterogeneous groups. This is understandable, since

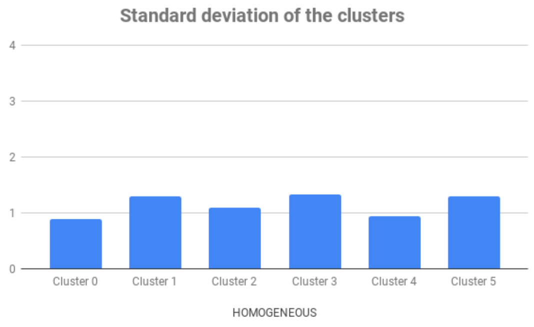


Figure 5.5: Standard deviation of the homogeneous clusters

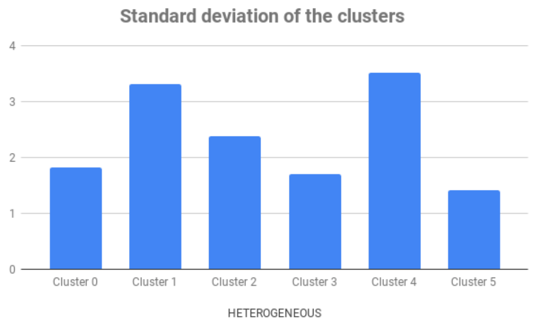


Figure 5.6: Standard deviation of the heterogeneous clusters

the means of students inside each homogeneous group are much similar to each other than the means inside an heterogeneous group, where the students have very different grades.

5.3 Experiments changing the weights of the course units

This experiment had the goal of showing how the weights in the course units can impact the profile of the groups created. In order to have results we could easily interpret, we used a dataset of fixed values, presented in Table 5.10.

As the table shows, all students have the same grades in course units UC2 and UC3, being UC1 the only that is different. When running the algorithm to form homogeneous groups with the same weight applied to all course units, the result will be influenced by the grades of UC2 and UC3, that define the best the profile of each student. The result of the clustering is shown in Table 5.11.

Now let's assume that the activity we want to group students for is highly related to unit course UC1. It would make sense to give a higher weight to this course unit than to the other two. So, we defined weights of 60%, 20% and 20% for course units UC1, UC2 and UC3, respectively. Running the algorithm again with these new parameters, the results were different, as described in Table 5.12.

Table 5.10: Dataset of 12 students generated by fixed values

	UC1	UC2	UC3
A	20	10	10
B	10	10	10
C	20	10	10
D	10	10	10
E	16	16	16
F	20	16	16
G	16	16	16
H	20	16	16
I	16	20	20
J	16	20	20
K	10	20	20
L	10	20	20

Table 5.11: Groups formed using the same weights for each course unit

	UC1	UC2	UC3	CLUSTER
A	20	10	10	cluster_0
B	10	10	10	cluster_0
C	20	10	10	cluster_0
D	10	10	10	cluster_0
E	16	16	16	cluster_1
F	20	16	16	cluster_1
G	16	16	16	cluster_1
H	20	16	16	cluster_1
I	16	20	20	cluster_2
J	16	20	20	cluster_2
K	10	20	20	cluster_2
L	10	20	20	cluster_2

Table 5.12: Groups formed using different weights for each course unit

	UC1	UC2	UC3	CLUSTER
A	20	10	10	cluster_1
B	10	10	10	cluster_0
C	20	10	10	cluster_1
D	10	10	10	cluster_0
E	16	16	16	cluster_2
F	20	16	16	cluster_1
G	16	16	16	cluster_2
H	20	16	16	cluster_1
I	16	20	20	cluster_2
J	16	20	20	cluster_2
K	10	20	20	cluster_0
L	10	20	20	cluster_0

It shows that the algorithm gave much more importance to the grades of course unit UC1 than the others, grouping the students according to that criteria.

5.4 Homogeneity inside clusters

An interesting metric also tested in the experiments was the homogeneity inside clusters. This was tested in homogeneous groups. Therefore, using the dataset of A.1, we ran the algorithm several times, with different values for *k*. Each time, for each cluster, we calculated the amplitude of the means of the cluster, given by the difference between the students of the cluster with the highest and lowest mean. Then, we calculated the mean of the amplitudes of all clusters to get the average amplitude of each run. The results can be consulted in Table 5.13. It shows that the homogeneity intra-cluster is inversely proportional to the value of *k*.

Table 5.13: Variation of the intra-cluster homogeneity with *k*

k	**3**	**4**	**6**	**8**	**12**
Amplitude	3.67	3.25	2.03	1.25	0.64

5.5 Conclusions

After doing the previous experiments, we can conclude that the algorithm fulfills its purpose. It can create groups in which the students have very similar grades and groups where the grades diverge the most. This is mostly visible in the last experiments, where the dataset is large and there is a higher probability of the students having different grades and means.

This said, it is also important to notice the relevance of the methods used for data generation. As seen in the experiments, not all methods allowed to conclude about the efficiency of the algorithm. In the case of the experiments done in Section 5.2.1, in which the data was generated according to 3.5.2, the results were inconclusive, because both methods of grouping (homogeneous and heterogeneous) returned similar results. This happened due to the lack of diversity in the data created, which makes us conclude that this methodology is somehow not useful in classes where the characteristics of the students are all very similar. However, that is not the case in most situations.

Another conclusion that can be made about the results is that, when creating heterogeneous groups, the results of the means of the cluster tend to be closer to the *mean* parameter defined in the data generation. In this specific case, all experiments were performed using a mean of 14, so the clusters' mean tend to be closer or equal to this value.

Concerning the weights of the course units, it can also be concluded that adding different weights effectively impacts the outcome of the results in the clustering process.

With respect to the homogeneity intra-cluster, in homogeneous groups, we concluded that the larger the group, the higher is its amplitude, since the means of the students, besides being similar, will diverge more. This means that, when clustering homogeneous groups, we should create groups with a small number of elements, in order to take advantage of the homogeneity of the group.

Finally, in terms of temporal efficiency of the algorithm, we present the runtimes of the several experiments done in this phase. They are shown in Table 5.14.

Table 5.14: Execution time of the algrithm with different input parameters

	Data	**k**	**Type**	**Time (ms)**
Fixed values	12	3	Homogeneous	49
	12	4	Heterogeneous	54
Normal distributions and cumulative probabilities	12	4	Homogeneous	77
	12	4	Heterogeneous	145
	24	12	Homogeneous	226
	24	12	Heterogeneous	384
	24	4	Homogeneous	344
	24	4	Heterogeneous	376
	24	6	Homogeneous	320
	24	6	Heterogeneous	563

As expected, the greater the dataset is, the greater is the running time of the algorithm. However, the table shows that even for the same dataset the runtimes are not that consistent, as seen in the last two experiments runtimes, for example. This can be due to the initial centroids selected. If the algorithm selects the worst centroids in the beggining, it will take much longer to converge to an optimum solution. On the other hand, if it choses the best points to be the centroids, it will reach the best solution very fast.

Also, we can conclude that there is no clear relationship between the type of groups to form and the runtime of the algorithm. Overall, it can be said that the runtime of the methodology is preety fast, since it can run in less than a second for groups of 24 students, independently of the number of groups to create.

Chapter 6

Conclusions and Future Work

6.1 Final conclusions

In this dissertation there was presented the problem of automatic group formation, within the scenario of effectively creating students groups in classes. The importance being given recently to collaborative learning in the field of Educations makes this a current problem. So, it is crucial to develop methodologies that aim to solve it.

In first place, we conclude that grouping students isn't a standard clustering problem. It is not as simple as grouping people by their similarities, since the groups depend on the type of activity to develop. Also, we should give importance to all the atributes that can be used to characterize a student. Not only the grades, but also more personal criteria, like preferences of the students. All characteristics can impact the final result, depending on their weight.

Also, we conclude that, besides having good attributes, it is very important to gather enough data to test the implementation. Specifically in our case, that wasn't possible, so we had to resort to data generation techniques. However, This data generated had a good approximation to reality, combinig normal distributions of the course units' grades and the means of students. It allowed us to perform good experiments and have interesting results.

About the methodology implemented, it can be said that we managed to develop a solid algorithm that is able to perform both types of clustering (homogeneous and heterogeneous) in very fast runtimes, for considerable amounts of data. The results obtained by it are what was expected, even when we defined diffrent weights for the attributes. The experiments conducted, along with the statistics obtained from the results corroborate this conclusion.

In general, we can conclude that this was a challenging but rewarding project, that can be very impactful in the future. Despite that, there is stil some work to be done, to ensure our methodology is ready to be used in a real environment.

6.2 Future Work

The first enhancement that can be done in the future is to add other important attributes (mentioned before) in addition to the grades, followed by a new experimentation phase to conclude about the importance of each attribute.

Another improvement that should be implemented in this project is to include a mechanism of reinforcement learning that is able to take the results of an activity and use them to help future processes of grouping. Also, the grouping process should not be static, but continuous instead, a students' preferences or grouping requirements can change with time.

Finally, after adding these two components, the methodology should be subjected to real life environments, to evaluate its efficiency in the task of grouping students.

References

[AA13] Raed T Aldahdooh and Wesam Ashour. DIMK-means "Distance-based Initialization Method for K-means Clustering Algorithm". *Intelligent Systems and Applications*, 02:41–51, 2013.

[ABRA+18] Ewa Andrejczuk, Rita Berger, Juan A. Rodriguez-Aguilar, Carles Sierra, and Víctor Marín-Puchades. *The composition and formation of effective teams: computer science meets organizational psychology*, volume 33. 2018.

[AMCM83] John R. (John Robert) Anderson, Ryszard S. (Ryszard Stanisław) Michalski, Jaime G. (Jaime Guillermo) Carbonell, and Tom M. (Tom Michael) Mitchell. *Machine learning : an artificial intelligence approach*. M. Kaufmann, 1983.

[AS95] Khaled S. Al-Sultan. A Tabu search approach to the clustering problem. *Pattern Recognition*, 28(9):1443–1451, 9 1995.

[Bha14] Aruna Bhat. K-medoids clustering using partitioning around medoids for performing face recognition. *International Journal of Soft Computing, Mathematics and Control*, 3(3):1–12, 2014.

[Del02] Frank Dellaert. The Expectation Maximization Algorithm. Technical report, 2002.

[Fle01] Neil D. Fleming. *Teaching and learning styles : VARK strategies*. Neil Fleming, 2001.

[Glo89] Fred Glover. Tabu Search - Part I. Technical Report 3, 1989.

[HFH+09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 11 2009.

[HFKB12] Kurt Hornik, Ingo Feinerer, Martin Kober, and Christian Buchta. Spherical k-Means Clustering. *Journal of Statistical Software*, 50(10):103–114, 2012.

[Hüb10] Roland Hübscher. Assigning students to groups using general and context-specific criteria. *IEEE Transactions on Learning Technologies*, 3(3):178–189, 2010.

[JD88] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice Hall, 1988.

[JH10] Xin Jin and Jiawei Han. *K-Medoids Clustering*, pages 564–565. Springer US, Boston, MA, 2010.

[JMF99] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 9 1999.

[KDHB17] Tina Knez, Martina Holenko Dlab, and Natasa Hoic-Bozic. Implementation of group formation algorithms in the ELARS recommender system. *International Journal of Emerging Technologies in Learning*, 12(11):198–207, 2017.

[KS10] Nobel Khandaker and Leen Kiat Soh. ClassroomWiki: A collaborative Wiki for instructional use with multiagent group formation. *IEEE Transactions on Learning Technologies*, 3(3):190–202, 2010.

[LSS10] Walter L Leite, Marilla Svinicki, and Yuying Shi. Attempted Validation of the Scores of the VARK: Learning Styles Inventory With Multitrait-Multimethod Confirmatory Factor Analysis Models. *Validity Studies Educational and Psychological Measurement*, 70(2):323–339, 2010.

[MKS$^+$15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518, 2015.

[MOW17] Elizaphan M. Maina, Robert O. Oboko, and Peter W. Waiganjo. Using Machine Learning Techniques to Support Group Formation in an Online Collaborative Learning Environment. *International Journal of Intelligent Systems and Applications*, 9(3):26–33, 2017.

[PKLMH96] Leslie Pack Kaelbling, Michael L Littman, Andrew W Moore, and Smith Hall. Reinforcement Learning: A Survey. Technical report, 1996.

[SB98] R.S. Sutton and A.G. Barto. Reinforcement Learning: An Introduction. *IEEE Transactions on Neural Networks*, 9(5):1054–1054, 9 1998.

[SB15] Ivan Srba and Maria Bielikova. Dynamic group formation as an approach to collaborative learning support. *IEEE Transactions on Learning Technologies*, 8(2):173–186, 2015.

[SKKR01] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the tenth international conference on World Wide Web - WWW '01*, pages 285–295, New York, New York, USA, 2001. ACM Press.

[SKS06] G. Stahl, T. Koschmann, and D. Suthers. Computer-supported collaborative learning: An historical perspective. *Cambridge handbook of the learning sciences*, pages 409–426, 2006.

[SM80] Ryszard S. Michalski. Knowledge Acquisition Through Conceptual Clustering: A Theoretical Framework and an Algorithm for Partitioning Data into Conjunctive Concepts. *Journal of Policy Analysis and Information Systems*, 4:219–244, 1980.

[SS15] Anurag Sarkar and Dibyabiva Seth. A New Approach to Collaborative Group Formation. 128(3):7–14, 2015.

[Stu] Students' Academic Performance Dataset | Kaggle.

[Sun07] Sung-Hyuk Cha. Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 1(4):300–307, 2007.

[Woo09] Michael J. Wooldridge. *An introduction to multiagent systems*. John Wiley & Sons, 2009.

REFERENCES

Appendix A

Experiments with 24 students

A.1 Dataset

Table A.1: Dataset of 24 students created with normal distributions and cumulative probabilities

	UC1	UC2	UC3	UC4	UC5	UC6
A	13	12	11	12	11	13
B	17	14	16	16	15	14
C	14	16	14	12	14	14
D	16	16	16	17	17	16
E	7	8	9	9	9	7
F	12	13	12	12	13	11
G	11	15	13	15	13	12
H	14	14	14	13	13	13
I	15	15	16	15	15	17
J	11	12	11	10	11	11
K	11	12	11	12	12	11
L	12	13	14	11	13	14
M	14	13	15	17	14	13
N	14	15	14	14	12	13
O	14	17	14	15	15	15
P	16	16	16	17	17	19
Q	14	14	16	15	15	15
R	10	11	10	10	9	10
S	15	15	16	15	16	16
T	12	11	10	10	11	9
U	16	13	13	13	14	14
V	15	15	13	15	14	14
W	19	17	18	18	19	19
X	16	14	15	15	15	14

A.2 Results of the students working in pairs

Table A.2: Results of the homogeneous groups created

	Homogeneous Groups											
Clusters	**Cluster 0**		**Cluster 1**		**Cluster 2**		**Cluster 3**		**Cluster 4**		**Cluster 5**	
Names	**G**	**M**	**U**	**V**	**D**	**S**	**F**	**L**	**H**	**N**	**B**	**X**
UC1	11	14	16	15	16	15	12	12	14	14	17	16
UC2	15	13	13	15	16	15	13	13	14	15	14	14
UC3	13	15	13	13	16	16	12	14	14	14	16	15
UC4	15	17	13	15	17	15	12	11	13	14	16	15
UC5	13	14	14	14	17	16	13	13	13	12	15	15
UC6	12	13	14	14	16	16	11	14	13	13	14	14

	Homogeneous Groups											
Clusters	**Cluster 6**		**Cluster 7**		**Cluster 8**		**Cluster 9**		**Cluster 10**		**Cluster 11**	
Names	**J**	**K**	**E**	**R**	**A**	**T**	**I**	**Q**	**P**	**W**	**C**	**O**
UC1	11	11	7	10	13	12	15	14	16	19	14	14
UC2	12	12	8	11	12	11	15	14	16	17	16	17
UC3	11	11	9	10	11	10	16	16	16	18	14	14
UC4	10	12	9	10	12	10	15	15	17	18	12	15
UC5	11	12	9	9	11	11	15	15	17	19	14	15
UC6	11	11	7	10	13	9	17	15	19	19	14	15

Table A.3: Results of the heterogeneous groups created

	Heterogeneous Groups											
Clusters	**Cluster 0**		**Cluster 1**		**Cluster 2**		**Cluster 3**		**Cluster 4**		**Cluster 5**	
Names	**R**	**S**	**G**	**U**	**A**	**B**	**I**	**K**	**C**	**M**	**L**	**Q**
UC1	10	15	11	16	13	17	15	11	14	14	12	14
UC2	11	15	15	13	12	14	15	12	16	13	13	14
UC3	10	16	13	13	11	16	16	11	14	15	14	16
UC4	10	15	15	13	12	16	15	12	12	17	11	15
UC5	9	16	13	14	11	15	15	12	14	14	13	15
UC6	10	16	12	14	13	14	17	11	14	13	14	15

	Heterogeneous Groups											
Clusters	**Cluster 6**		**Cluster 7**		**Cluster 8**		**Cluster 9**		**Cluster 10**		**Cluster 11**	
Names	**H**	**V**	**N**	**X**	**E**	**W**	**F**	**O**	**D**	**J**	**P**	**T**
UC1	14	15	14	16	7	19	12	14	16	11	16	12
UC2	14	15	15	14	8	17	13	17	16	12	16	11
UC3	14	13	14	15	9	18	12	14	16	11	16	10
UC4	13	15	14	15	9	18	12	15	17	10	17	10
UC5	13	14	12	15	9	19	13	15	17	11	17	11
UC6	13	14	13	14	7	19	11	15	16	11	19	9

Experiments with 24 students

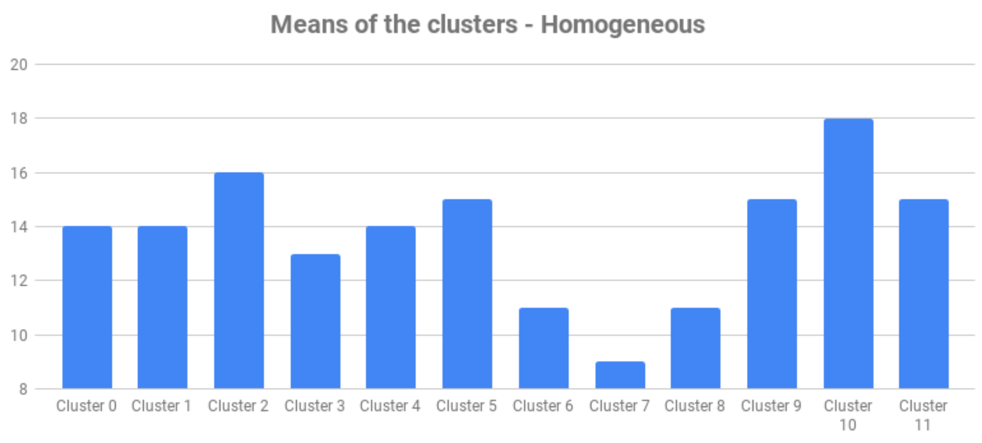


Figure A.1: Means of the homogeneous clusters

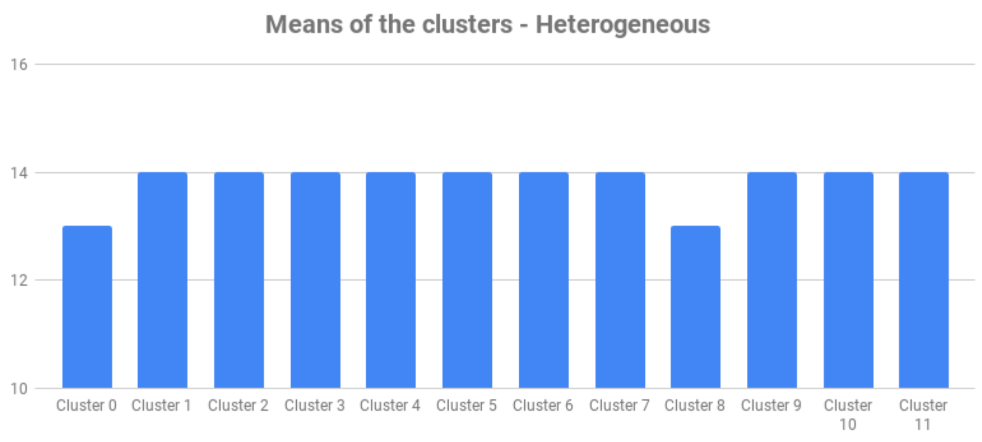


Figure A.2: Means of the heterogeneous clusters

A.3 Results of the students working in groups of 6

Table A.4: Results of the homogeneous groups created

	Homogeneous Groups											
Clusters	**Cluster 0**						**Cluster 1**					
Names	**C**	**F**	**H**	**L**	**M**	**N**	**B**	**D**	**I**	**P**	**Q**	**S**
UC1	14	12	14	12	14	14	17	16	15	16	14	15
UC2	16	13	14	13	13	15	14	16	15	16	14	15
UC3	14	12	14	14	15	14	16	16	16	16	16	16
UC4	12	12	13	11	17	14	16	17	15	17	15	15
UC5	14	13	13	13	14	12	15	17	15	17	15	16
UC6	14	11	13	14	13	13	14	16	17	19	15	16

	Homogeneous Groups											
Clusters	**Cluster 2**						**Cluster 3**					
Names	**A**	**E**	**J**	**K**	**R**	**T**	**G**	**O**	**U**	**V**	**W**	**X**
UC1	13	7	11	11	10	12	11	14	16	15	19	16
UC2	12	8	12	12	11	11	15	17	13	15	17	14
UC3	11	9	11	11	10	10	13	14	13	13	18	15
UC4	12	9	10	12	10	10	15	15	13	15	18	15
UC5	11	9	11	12	9	11	13	15	14	14	19	15
UC6	13	7	11	11	10	9	12	15	14	14	19	14

Table A.5: Results of the heterogeneous groups created

	Heterogeneous Groups											
Clusters	**Cluster 0**						**Cluster 1**					
Names	**G**	**H**	**L**	**N**	**O**	**U**	**C**	**M**	**Q**	**T**	**V**	**X**
UC1	11	14	12	14	14	16	14	14	14	12	15	16
UC2	15	14	13	15	17	13	16	13	14	11	15	14
UC3	13	14	14	14	14	13	14	15	16	10	13	15
UC4	15	13	11	14	15	13	12	17	15	10	15	15
UC5	13	13	13	12	15	14	14	14	15	11	14	15
UC6	12	13	14	13	15	14	14	13	15	9	14	14

	Heterogeneous Groups											
Clusters	**Cluster 2**						**Cluster 3**					
Names	**B**	**D**	**E**	**I**	**P**	**S**	**A**	**F**	**J**	**K**	**R**	**W**
UC1	17	16	7	15	16	15	13	12	11	11	10	19
UC2	14	16	8	15	16	15	12	13	12	12	11	17
UC3	16	16	9	16	16	16	11	12	11	11	10	18
UC4	16	17	9	15	17	15	12	12	10	12	10	18
UC5	15	17	9	15	17	16	11	13	11	12	9	19
UC6	14	16	7	17	19	16	13	11	11	11	10	19

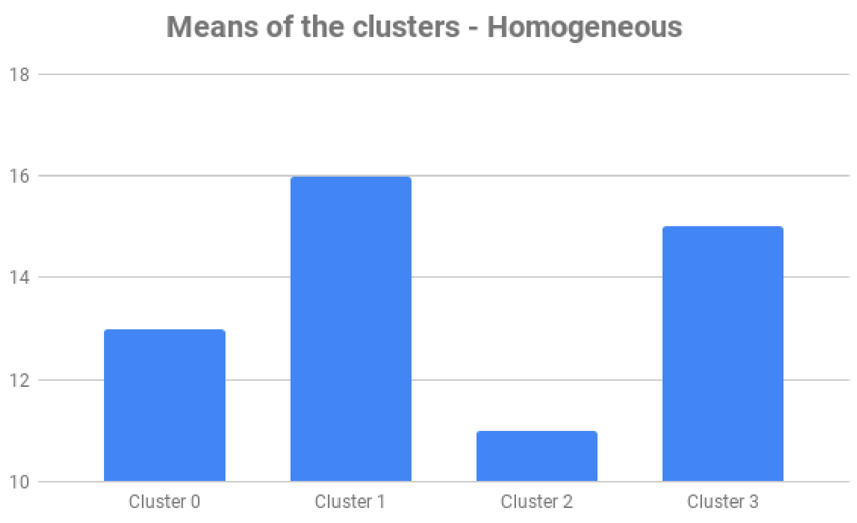


Figure A.3: Means of the homogeneous clusters

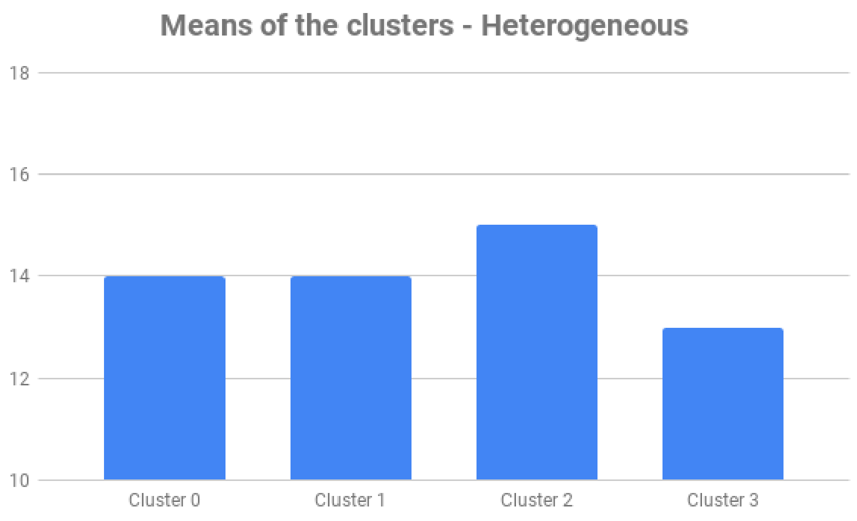


Figure A.4: Means of the heterogeneous clusters

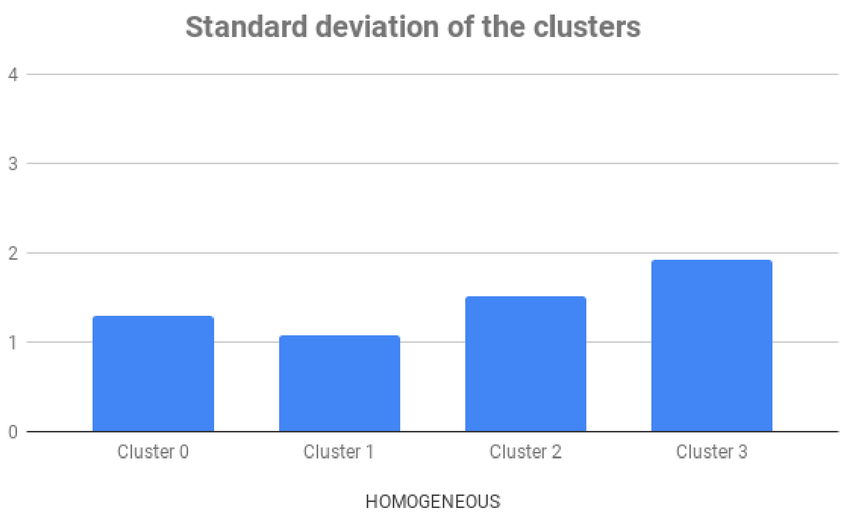


Figure A.5: Standard deviation of the homogeneous clusters

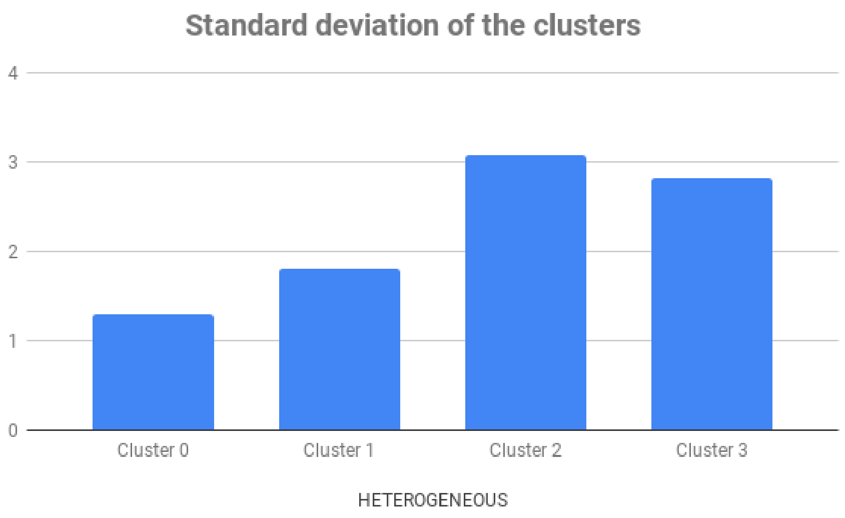


Figure A.6: Standard deviation of the heterogeneous clusters

A.4 Results of the students working in groups of 4

Table A.6: Results of the homogeneous groups created

	Homogeneous Groups											
Clusters	**Cluster 0**				**Cluster 1**				**Cluster 2**			
Names	**B**	**I**	**Q**	**X**	**D**	**P**	**S**	**W**	**C**	**H**	**N**	**O**
UC1	17	15	14	16	16	16	15	19	14	14	14	14
UC2	14	15	14	14	16	16	15	17	16	14	15	17
UC3	16	16	16	15	16	16	16	18	14	14	14	14
UC4	16	15	15	15	17	17	15	18	12	13	14	15
UC5	15	15	15	15	17	17	16	19	14	13	12	15
UC6	14	17	15	14	16	19	16	19	14	13	13	15

	Homogeneous Groups											
Clusters	**Cluster 3**				**Cluster 4**				**Cluster 5**			
Names	**E**	**J**	**R**	**T**	**A**	**F**	**K**	**L**	**G**	**M**	**U**	**V**
UC1	7	11	10	12	13	12	11	12	11	14	16	15
UC2	8	12	11	11	12	13	12	13	15	13	13	15
UC3	9	11	10	10	11	12	11	14	13	15	13	13
UC4	9	10	10	10	12	12	12	11	15	17	13	15
UC5	9	11	9	11	11	13	12	13	13	14	14	14
UC6	7	11	10	9	13	11	11	14	12	13	14	14

Table A.7: Results of the heterogeneous groups created

	Heterogeneous Groups											
Clusters	**Cluster 0**				**Cluster 1**				**Cluster 2**			
Names	**J**	**M**	**Q**	**V**	**A**	**F**	**T**	**W**	**B**	**O**	**R**	**X**
UC1	11	14	14	15	13	12	12	19	17	14	10	16
UC2	12	13	14	15	12	13	11	17	14	17	11	14
UC3	11	15	16	13	11	12	10	18	16	14	10	15
UC4	10	17	15	15	12	12	10	18	16	15	10	15
UC5	11	14	15	14	11	13	11	19	15	15	9	15
UC6	11	13	15	14	13	11	9	19	14	15	10	14

	Heterogeneous Groups											
Clusters	**Cluster 3**				**Cluster 4**				**Cluster 5**			
Names	**D**	**G**	**H**	**L**	**E**	**I**	**P**	**S**	**C**	**K**	**N**	**U**
UC1	16	11	14	12	7	15	16	15	14	11	14	16
UC2	16	15	14	13	8	15	16	15	16	12	15	13
UC3	16	13	14	14	9	16	16	16	14	11	14	13
UC4	17	15	13	11	9	15	17	15	12	12	14	13
UC5	17	13	13	13	9	15	17	16	14	12	12	14
UC6	16	12	13	14	7	17	19	16	14	11	13	14

Experiments with 24 students

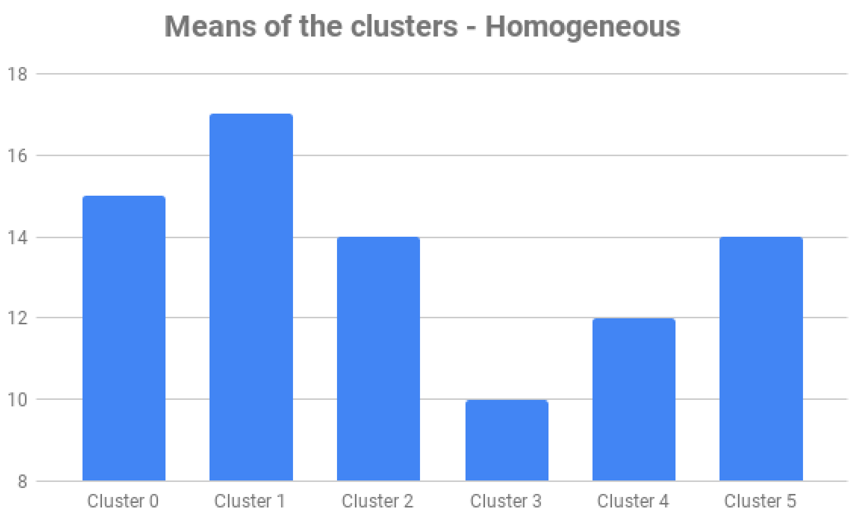


Figure A.7: Means of the homogeneous clusters

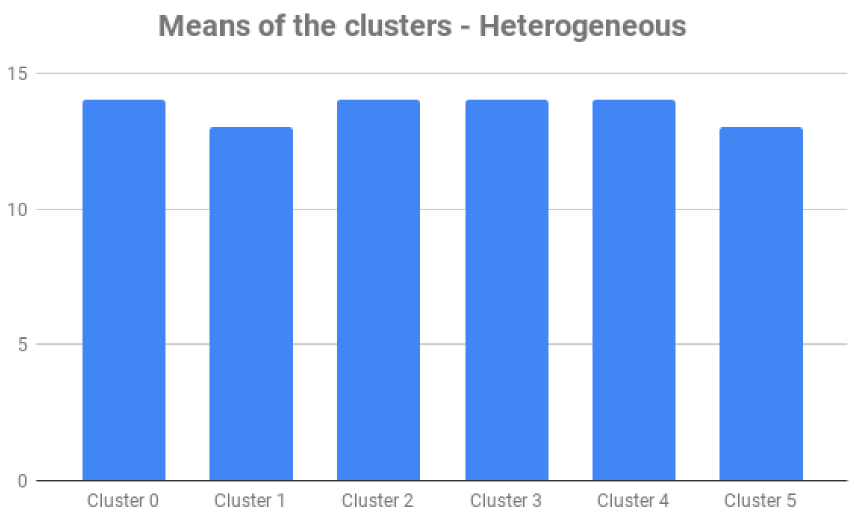


Figure A.8: Means of the heterogeneous clusters

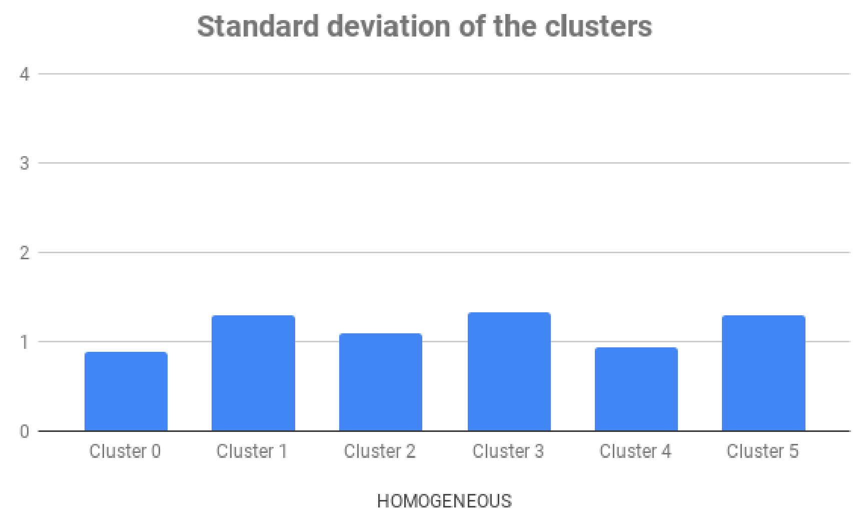


Figure A.9: Standard deviation of the homogeneous clusters

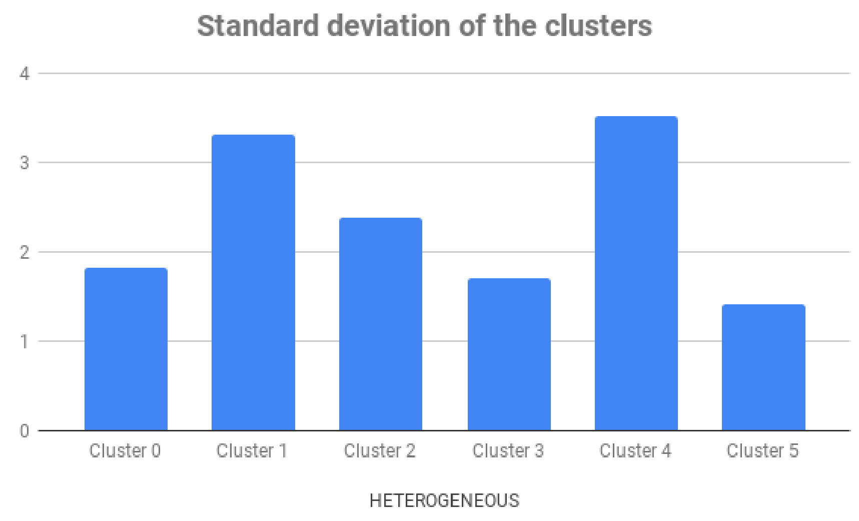


Figure A.10: Standard deviation of the heterogeneous clusters